Modellierung und Optimierung von mehrstufigen Umschlagsystemen

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Maschinenbau
der Universität Karlsruhe (TH)
genehmigte

Dissertation

von

Dipl.-Ing. Knut Alicke

aus Worms

Tag der mündlichen Prüfung: 25. Oktober 1999
Hauptreferent: Prof. Dr.-Ing. Dr. h.c. D. Arnold
Korreferent: Prof. Dr. H.-O. Günther

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Fördertechnik und Logistiksysteme der Universität Karlsruhe (TH).

Herr Prof. Dr.-Ing. Dr. h.c. Dieter Arnold, dem Lehrstuhlinhaber und Leiter des Instituts für Fördertechnik und Logistiksysteme, hat mein Interesse für die Optimierung von Logistiksystemen geweckt. Ihm gilt mein besonderer Dank für die Übernahme des Hauptreferates und die stete Bereitschaft, über Modelle und neue Ansätze zu diskutieren. Seine offene und vertraute Art hat mich nachhaltig beeinflußt und die nötigen Freiräume geschaffen, um erfolgreich auf dem Gebiet der Optimierung von Logistiksystemen zu forschen und zu arbeiten.

Herrn Prof. Dr. Hans-Otto Günther, Lehrstuhlinhaber des Fachgebiets Produktionsmanagement der Technischen Universität Berlin danke ich sehr herzlich für die Übernahme des Korreferates. Seine wertvollen Anregungen haben der Arbeit den letzten Schliff gegeben.

Meinen Kollegen danke ich für die zahlreichen Diskussionen, die schonungslose Kritik und den Spaß am Lösen von praktischen sowie theoretischen Problemen. Die Atmosphäre im Forschungsbereich Materialfluß und Logistik hat wesentlich zum Gelingen dieser Arbeit beigetragen.

Meinen Eltern danke ich, daß sie mir die Ausbildung und damit den Grundstein zu dieser Arbeit ermöglicht haben.

Mein spezieller Dank gilt meiner Lebensgefährtin und schärfsten Kritikerin Sabine; denn was bedeuten schon zwei Buchstaben vor dem Namen im Vergleich zu ihrer Liebe und Zuneigung.

Karlsruhe, im Herbst 1999 Knut Alicke

Kurzfassung

Alicke, Knut
Modellierung und Optimierung von mehrstufigen Umschlagsystemen

Der mehrstufige Umschlag von Gütern, etwa in großen Distributionszentren oder Güterverteilzentren, gewinnt an Bedeutung. Um den Servicegrad eines solchen mehrstufigen Umschlagsystems zu verbessern, müssen komplexe Planungs- und Steuerungsprobleme gelöst werden. Entscheidend ist hierbei die Bestimmung einer optimalen Umschlagsreihenfolge.

In dieser Arbeit werden spezielle sog. *Constraints* zur Modellierung von mehrstufigen Umschlagsystemen entwickelt, Elemente des Systems eingeführt und Beziehungen zwischen diesen Elementen definiert. Anschließend werden Ansätze des Constraint Satisfaction benutzt, um die Systeme in ein Optimierungsmodell zu überführen. Die Zeit, die für die Suche der (sub-) optimalen Lösung benötigt wird, ist ausreichend für eine praktische Anwendung des Verfahrens. Weiterhin werden neue Ansätze zur effizienten Lösungssuche und zur Einschränkung des Suchraumes entwickelt und benutzt.

Das Verfahren wird auf ein Kommissioniersystem angewendet, wie es in großen Distributionszentren zu finden ist, und auf ein Container-Umschlagsterminal, das prototypisch im kombinierten Ladungsverkehr eingesetzt wird. Bei dem Kommissioniersystem sind begrenzte Puffer zwischen den Stufen, ein Personal-Pool sowie spezielle Reihenfolgebedingungen zu beachten. Bei dem Umschlagterminal sind insbesondere die reihenfolgeabhängigen Leerfahrten der Kräne, die alternativen Zuordnungen von Containern zu Kränen und die Verwendung einer Ressource, die zugleich eine puffernde und bearbeitende Funktion ausübt, von Interesse.

Die Ergebnisse belegen die Anwendbarkeit des Verfahrens auf praktische Probleme und geben neue Impulse für die zukünftige Forschung auf diesem Gebiet.

Abstract

Alicke, Knut
Modeling and Optimization of multi-stage transshipment systems

The importance of multi-stage transshipment of goods, for example in large distribution warehouses, is raising. To increase the service-level of such systems, the underlying planning- and control problems have to be solved. The aim is to determine an optimal transshipment-sequence.

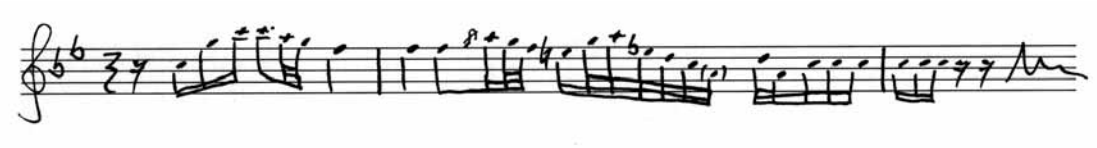
In this dissertation, special so called *constraints* are developed to model mult-stage transshipment systems, elements of the system are introduced and dependencies, the constraints, between the elements are defined. Afterwards Constraint Satisfaction is used to transform the system into an optimization model. The time to determine the (sub-)optimal solution is sufficient for a practical application of the method. New approaches for an efficient search for a solution and reduction of the search space are developed and applied.

The approach is applied to an order picking system, as it is found in large distribution warehouses and a rail-rail container-terminal, which is used as a prototype in the combined traffic. In the order picking system, limited buffers between the stages, a pool of personal and special sequence-constraints have to be considered for the order picking system. At the container terminal, especially the sequence dependent idle moves of the cranes, the alternative assignment of containers to cranes and the usage of a resource which combines buffering and operating is of major interest.

The results prove the applicability to practical problems and are showing new directions for future research in this area.

Inhaltsverzeichnis

1. Einleitung



Diese Arbeit beschäftigt sich mit Reihenfolgen in logistischen Systemen. In der Musik, wie an dem Ausschnitt eines Solos von Bob Berg[1] deutlich wird, ist die Reihenfolge der Töne wichtig. Durch ein Umstellen der Noten verliert die Phrase an Ausdruck, bis sie sogar falsch klingt. Im Vergleich zu technischen Systemen ist die Bewertung eines solchen musikalischen Fragmentes äußerst schwierig. Was ist optimal, also das Beste, die funktionsharmonische Reinheit oder das spontane Interagieren mit den Mitmusikern in der Improvisation des Augenblicks. Die Improvisation ist auch im Hinblick auf technische Systeme sehr interessant. Hier wird spontan komponiert, es entsteht nicht mehr veränderbare Musik[2]. Eine Improvisation ist keine totale Neukomposition, Musiker entwickeln häufig einen eigenen Stil. Es handelt sich vielmehr um ein spontanes Reagieren auf Basis eines Vorrates musikalischer Erfahrung, d.h. Patterns, Stile und funktionsharmonischer Zusammenhänge. Auch in technischen Systemen werden Entscheidungen in Echtzeit getroffen, allerdings fließt hier zum Zeitpunkt der Entscheidung nur ein Bruchteil des Wissens über die modellierten Systeme ein, der Entscheidungsprozeß wäre viel zu langwierig.

Neben der spontanen, parallelen Entscheidung über Töne und Rhythmus ist auch die Darstellung in Notenschrift bemerkenswert. Informationen über Tonhöhe und Rhythmus fließen in mehreren parallelen Ebenen zusammen. Durch Standardisierung und Training haben Fremde eine gemeinsame Basis zum Musizieren. In

[1] Amerikanischer Saxophonist, spielte u.a. bei Miles Davis. Das Solofragment stammt aus *Words* auf der CD *Short Stories*.

[2] Der darstellende Künstler hat im Gegensatz zum Musiker den Vorteil (oder Nachteil), an dem Kunstwerk nachträgliche Veränderungen vornehmen zu können.

technischen Systemen stellt beispielsweise ein Gantt-Chart ein ähnliches Konstrukt dar, um zeitliche Abläufe von (parallelen) Vorgängen zu visualisieren.

Die Wirkung von Musik entfaltet sich im Ganzen, also holistisch, das Weglassen von Instrumenten oder Passagen zerstört den Eindruck. Ähnliches geschieht bei der Planung und Verbesserung von technischen Systemen, wo der holistische Gedanke allzu oft vernachlässigt wird. Durch die Vertiefung in Details geht der Blick für die (globalen) Zusammenhänge verloren, es wird lokal anstatt global optimiert.

In dieser Arbeit wird ein holistischer Ansatz vorgestellt, um mehrstufige Umschlagsysteme zu modellieren und zu optimieren. Die Idee und Motivation hierzu kam mir bei zahlreichen Projekten, die ich am Insitut für Fördertechnik und Logistiksysteme bearbeitete. In Distributionszentren, die in Kapitel 6 behandelt werden, stellten sich immer wieder überlaufende Puffer und schlecht ausgelastetes Personal als Problem heraus. Das in Kapitel 7 beschriebene und optimierte Umschlagsterminal für den kombinierten Ladungsverkehr, der „Mega Hub“, ist die Grundlage für ein sehr komplexes und daher interessantes Optimierungproblem.

Schlagwörter wie „Logistik ersetzt Bestände durch Information!“ gehen durch die Presse. Doch woher stammt diese bestandsverringernde Information? Unter anderem kann sie aus Optimierungsmodellen und Verfahren abgeleitet werden. Um die theoretischen Verfahren auf praktische Problem anzuwenden, ist es wichtig, zielgerichtete Annahmen und Vereinfachungen zu treffen um spekulative und stochastische Bestände so weit als möglich zu vermeiden. Die beiden Anwendungskapitel 6 und 7 zeigen, wo sinnvolle Grenzen zu ziehen sind.

2. Zielsetzung und Vorgehen

The shortest distance from research to application.

Die Improvisation zu formalisieren, anzuwenden und nachzuvollziehen, quantitative Methoden auf schwierige, praxisrelevante Problemstellungen anzuwenden, die Systeme besser zu verstehen und zu optimieren; in einem komplexen Umfeld schnell zu agieren und die Lücke zwischen theoretischen Arbeiten und praktischen Anwendungen zu schließen; all diese Ziele sollen in den nächsten Kapiteln verfolgt werden.

Die praxisnahen Modellierung und Optimierung von sehr komplexen Systemen wird in der Literatur wenig behandelt. Bei vielen Arbeiten werden Annahmen und Vereinfachungen getroffen, die eine Übertragbarkeit der Ergebnisse in die Praxis zweifelhaft erscheinen lassen. Aus diesem Grund ist das Ziel der vorliegenden Arbeit die *praxisnahe Modellierung* und konsequenterweise auch die *Optimierung* von *komplexen*, *mehrstufigen* Umschlagsystemen.

Wie werden Systeme geplant, verbessert, optimiert? Zuerst muß das System mit den enthaltenen Elementen und deren wechselseitigen Beziehungen verstanden werden. In dem Schritt der *Modellierung* werden die identifizierten Elemente mit ihren Beziehungen in ein abstraktes Modell überführt. Das kann in einer formalen oder umgangssprachlichen Weise geschehen. Das verfolgte Ziel ist für das weitere Vorgehen sehr wichtig, d.h. sollen möglichst viele der Restriktionen erfüllt werden, oder soll der Wert einer Variablen (max. Verspätung) minimiert werden? Zur Lösung des Problems ist ein (konkretes) Verfahren nötig, das Modell wird mittels der *Formulierung* auf dieses Verfahren übertragen. Um das Problem zu lösen, wird die Formulierung in einem rechnerverständlichen Code *implementiert*. Sehr wichtig für die Anwendbarkeit ist die *Robustheit* eines implementierten Modelles. Unter robust soll hier verstanden werden, daß bei unterschiedlichen Ausgangsparametern (aber gleicher Problemkomplexität) eine Lösung in ähnlicher Zeit und Güte ermittelt wird. Mit einem Experimentendesign kann das Verhalten des Systems systematisch untersucht und die Sensitivität überprüft werden.

Nuijten (1994) unterscheidet Methoden des *Operations Research* (OR) und der *Artificial Intelligence* (AI): OR fokussiert die Lösung sehr spezieller, sehr schwerer Probleme; AI dagegen versucht, die Probleme mit Hilfe von allgemeineren Modellen zu lösen, evtl. mit Einbußen in der Laufzeit. Die in dieser Arbeit verwendeten AI-Ansätze bieten durch das Konzept der auf einer beliebig detaillierten Stufe relaxierbaren Constraints[1] eine sehr praxisnahe Modellierung.

Die strikte Trennung zwischen OR und AI läßt sich heute nicht mehr aufrechterhalten. Ein Beispiel ist die Entwicklung und Anwendung von *Meta-Heuristiken*, die ebenfalls nicht nur spezielle Probleme lösen. Nach Osman und Kelly (1996a) führt die Meta-Heuristik eine untergeordnete Heuristik durch eine intelligente Suche und Auswertung des Lösungsraumes zu einer (sub-)optimalen Lösung. D.h. hier werden Ansätze des AI und OR kombiniert verwendet. Auch im Bereich des hier angewendeten Constraint Satisfaction werden unterschiedliche Methoden des OR und AI kombiniert, denn das Ziel ist die Bestimmung einer (sub-)optimalen Lösung in möglichst kurzer Zeit.

Dieses Konzept wird in der vorliegenden Arbeit auf die Modellierung und Optimierung mehrstufiger Umschlagsysteme angewendet. Die Grundlagen der Modellierung und der Lösungsmethodik des ursprünglichen (AI-) Verfahrens Constraint Logic Programming werden geschaffen und anschließend verwendet, um zwei praxisrelevante Probleme zu lösen.

2.1. Aufbau der Arbeit

Die Improvisation stand nicht Pate bei der vorliegenden Arbeit, eher das im Vorfeld nötige Üben, durch Verstehen und Analysieren von formalen Strukturen. So werden in Kapitel 3 bestehende Ansätze aus dem Bereich des *Scheduling* vorgestellt. Arbeiten aus dem relativ jungen Forschungsgebiet des Constraint Programming werden anschließend beschrieben. Ähnliche Probleme wie bei dem untersuchten Mega Hub existieren z.B. bei sog. *Hoist Scheduling Problems*, die ebenfalls aufgezeigt werden. Ansätze zur Lösung von Teilproblemen bei der Optimierung von mehrstufigen Kommissioniersystemen werden vorgestellt. Dem Autor sind keine ganzheitlichen Ansätze in diesem Bereich bekannt. Wichtig ist die Kritik an theoretischen Verfahren, die durchweg von renommierten Forschern stammt.

[1] In der vorliegenden Arbeit wird durchgehend der Begriff *constraint* verwendet, der sich mit „Einschränkung“ oder bei Optimierungsmodellen mit „Restriktion“ übersetzen läßt. Auch weitere, englische Begriffe werden nicht übersetzt, um keine unnötige neue Begriffswelt zu definieren. So bleibt der Bezug zu der existierenden, englischsprachigen Fachliteratur gegeben.

Im Kapitel 4 wird das formale Gerüst geschaffen, um mehrstufige Systeme zu modellieren. Der Begriff des Systems wird eingeführt, die Elemente und die Beziehungen untereinander werden definiert, sowie temporale und kapazitative Constraints eingeführt. Ressourcen werden in bearbeitende, puffernde oder Kombinationen von bearbeitenden und puffernden unterteilt, es werden Constraints für unterschiedliche Übergangszeiten zwischen Operationen eingeführt.

Die Modellierung wird nun in Kapitel 5 verwendet, um ein Constraint Optimization Problem (COP) zu formulieren und mit Ansätzen des Constraint Logic Programming (CLP) zu lösen. Nachdem mit der Formulierung eines Constraint Satisfaction Problems (CSP) die theoretische Basis geschaffen ist, werden die speziellen Bedingungen zur Optimierung eines mehrstufigen Umschlagsystems integriert. Bedingungen zur Prüfung und Sicherstellung von Konsistenz werden vorgestellt, das Konzept des Backtracking wird eingeführt. Anschließend werden unterschiedliche Ansätze zum Suchen (und Finden) von Lösungen entwickelt und aufgezeigt, wie mit Hilfe von CLP ein Problem optimiert werden kann.

Die Anwendungen zeigen das Vorgehen an praktischen Problemen auf. Das erste Beispiel ist ein mehrstufiges Kommissioniersystem, das in Kapitel 6 modelliert und formuliert wird. Hier tritt das interessante Problem auf, daß eine *out-tree* in eine *in-tree* Struktur übergeht. Weiterhin werden neue reihenfolgeabhängige Übergangszeiten und die Beachtung eines Personal-Pools integriert. Die Lösung der simultanen Batchbildung und Reihenfolgebestimmung über mehrere Stufen wird vorgestellt und das Ergebnis diskutiert.

Die Modellierung und Optimierung eines intermodalen Umschlagterminals, des Mega Hub, wird in Kapitel 7 beschrieben. Hier liegt ein Problem mit alternativen Ressourcen, reihenfolgeabhängigen Übergängen und reihenfolgeabhängiger Anzahl der Operationen, sowie beliebigen Flüssen durch das System vor. Die Anwendung auf ein reales System wird vorgestellt, und die Ergebnisse werden bewertet.

Nach der Anwendung der theoretischen Ansätze auf praktische Probleme werden einige Erweiterungen und Richtungen für zukünftige Forschungen in Kapitel 8 als Ausblick aufgezeigt. Schließlich faßt das Kapitel 9 die Arbeit zusammen.

2.2. Nutzen für den Leser

Die Anwendungen von Constraint Logic Programming nehmen zu. Trotzdem gibt es bisher keine Unterscheidung zwischen puffernden und bearbeitenden Ressourcen mit der entsprechenden Modellierung. Die hieraus abgeleiteten Bedingungen

für die Konsistenz-Prüfung bzw. Sicherstellung ermöglichen die Anwendung auf mehrstufige Umschlagsysteme.

Die Anwendung auf Probleme aus recht unterschiedlichen Gebieten hat gezeigt, daß sich viele praktische Probleme auf eine ähnliche Art abstrahieren und modellieren lassen. Durch diese gemeinsame Struktur werden auch allgemeine Lösungsverfahren anwendbar. Es ist nicht notwendig, für jedes Problem eine neue Modellierung und einen neuen Lösungsweg zu entwickeln, sondern es kann auf bestehende Arbeiten zurückgegriffen werden.

In den beiden untersuchten Anwendungsfällen wurden bisher nur in sehr begrenztem Umfang Optimierungsverfahren eingesetzt, auch Anwendungen von Constraint Logic Programming sind hierzu bislang nicht veröffentlicht.

Die Ergebnisse bestätigen, daß durch die praxisnahe Erweiterung von theoretischen Verfahren die Anwendbarkeit gegeben ist. Allerdings ist deutlich geworden, daß die Komplexität von praktischen Problemen durch Hinzufügen von Randbedingungen beliebig wachsen kann. Die Lücke zwischen dem praktischen Einsatz und der theoretischen Entwicklung von Verfahren wurde verringert.

3. Stand der Forschung

Dieses Kapitel gibt einen Überblick über die aktuelle Forschung auf dem Gebiet der mehrstufigen (Umschlag-)Systeme. Hier spielen viele verwandte Bereiche mit hinein, die nicht erschöpfend behandelt werden. An geeigneten Stellen wird jedoch auf Übersichtsbeiträge und weiterführende Literatur verwiesen.

In Kapitel 3.1 werden die für mehrstufige Umschlagsysteme interessanten Ansätze aus dem Bereich des Scheduling vorgestellt. Die Anwendung von Constraint Satisfaction Ansätzen auf Optimierungs-Probleme wird in Kapitel 3.2 gezeigt. Im Bereich des Crane-Scheduling, hier bei sog. Hoist-Scheduling-Problemen, sind Probleme zu finden, die dem des Mega Hub ähnlich sind. Diese werden in Kapitel 3.4 beschrieben.

Die Modellierung und Optimierung von mehrstufigen Kommissioniersystemen beschränkt sich auf Teilsysteme; wichtige Ansätze werden in Kapitel 3.3 beschrieben. Das Spannungsfeld von theoretischen Annahmen und praktischen Anforderungen wird in Kapitel 3.5 dargelegt. Mit einer Zusammenfassung schließt das Kapitel ab.

3.1. Scheduling

Ein Teil des Operations Research/Management Science (OR/MS) beschäftigt sich mit den Fragen nach einer optimalen Ressourcenbelegung - dem *Scheduling*. Es ist eine Menge von Aufträgen gegeben, die wiederum aus Operationen bestehen. Für die Aufträge sind frühest mögliche Startzeitpunkte, gewünschte Fertigstellungstermine und die Bearbeitungsreihenfolge der enthaltenen Operationen gegeben. Das Ziel der Ansätze ist es, für diese Menge von Aufträgen einen Belegungsplan zu ermitteln, der bestands-, termin- oder auslastungsorientierte Ziele erfüllt. Ist die Ressourcen-Folge für alle Aufträge gleich, spricht man von einem *flow shop*-, andernfalls von einem *job shop*-Problem. Unterschieden wird weiterhin nach *einstufigen* und *mehrstufigen* Problemen, die wiederum *deterministisch* und *stochastisch* sein können. Auf jeder Stufe kann eine *einzelne* Ressource oder mehrere

parallele Ressourcen angeordnet sein. Mehrere Ressourcen sind in der Regel über *Puffer* entkoppelt. Können unterschiedliche Ressourcen für die Bearbeitung eines Auftrages *alternativ* verwendet werden, lassen sich diese nach Dörrsam (1999) weiter klassifizieren in zeit- und funktions-*homoge* bzw. -*inhomoge* Maschinengruppen. Werden mehrere Aufträge auf einer Ressource bearbeitet, sind häufig *reihenfolgeabhängige Rüstzeiten* zu beachten.

Eine allgemeine Klassifizierung der im Kontext von Scheduling zu lösenden Probleme wird von Derrick und Pegman (1998) gegeben:

- *Scheduling*: Bestimmen von Startzeitpunkten für alle einzuplanenden Operationen.

- *Ressourcen-Allokation*: Zuweisung von Aktivitäten zu Ressourcen, so daß die Nachfrage befriedigt und gegebene Kapazitätsgrenzen nicht überschritten werden.

- *Reihenfolgeplanung*: Bestimmung einer Ordnung der Operationen, ohne die konkreten Startzeitpunkte zu berechnen.

- *Mischformen*: In praxisrelevanten Fragestellungen treten oft mehrere der Probleme zugleich auf, beispielsweise wenn Ressourcen unterschiedlicher *Qualifikation* (Alter, Geschwindigkeit, Toleranz, etc.) vorhanden sind, um eine Operation auszuführen. Zur Einplanung einer Operation sollte gleichzeitig eine Ressource aus der Menge der möglichen, *alternativen* Ressourcen ausgewählt und ein Startzeitpunkt bestimmt werden. Die Autoren kritisieren Ansätze, die solche kombinierten Probleme getrennt voneinander betrachten und lösen, d.h. zuerst die Allokation der Operation zu den Ressourcen mit anschließendem Scheduling oder umgekehrt.

In den vergangenen Jahren wurden zahlreiche Arbeiten im Bereich des Scheduling veröffentlicht. Eine frühe Darstellung ist in Baker (1974) und Ashour (1972) gegeben. Eine sehr umfassende Beschreibung der Problematik und bestehender Lösungsverfahren gibt Pinedo (1995). In dem sehr umfangreichen Artikel von Blazewicz, Domschke und Pesch (1996) werden die einzelnen Problemklassen dargestellt und exakte (Branch&Bound) sowie approximative Algorithmen beschrieben. Unter die letztgenannten fallen Prioritätsregeln, die sog. Shifting Bottleneck Heuristic, Opportunistic Scheduling, lokale Suchverfahren (Evolutionäre Verfahren, Tabu-Search, Simulated Annealing) und Constraint Propagation. Job-Shop

Probleme gehören zu der Klasse der NP Probleme[1] und zählen damit nach Graves, Rinnooy Kan und Zipkin (1993b) zu den schwierigsten untersuchten Problemen der kombinatorischen Optimierung.

Job-Shop Probleme werden mit Hilfe der Notation $n \times m$ klassifiziert. Die Anzahl der Aufträge ist hierbei n, die Anzahl der Ressourcen m. Das bekannte und immer wieder untersuchte 10-Aufträge, 10-Maschinen-Problem (10×10) wurde beispielsweise von Fischer und Thompson (1963) modelliert, konnte aber erst 1989 von Carlier und Pinson (1989) durch implizite Enummeration optimal gelöst werden. Es würde den Rahmen dieser Arbeit sprengen, die zahlreichen Verfahren, die sich oft nur in Nuancen unterscheiden, zur exakten oder approximativen Lösung von Job-Shop Problemen vorzustellen. Der interessierte Leser sei daher auf den erwähnten Artikel von Blazewicz, Domschke und Pesch (1996) verwiesen.

Die Mehrzahl der Scheduling-Ansätze vernachlässigt Rüstzeiten oder integriert sie in die Bearbeitungszeit. Eine Rüstzeit ist jedoch oftmals reihenfolgeabhängig. Es kann beispielsweise aufwendiger sein, einen Farbwechsel von Schwarz nach Weiß durchzuführen als umgekehrt. Die alleinige Beachtung der Rüstkosten, ohne die Rüstzeiten in eine Nebenbedingung zu integrieren, führt zu unzulässigen Lösungen, da die Kapazität der verwendeten Ressourcen systematisch *überschätzt* wird.

Die Rüstkosten können reduziert werden, wenn nicht bedarfssynchron gefertigt wird, sondern mehrere Aufträge zu Losen zusammengefaßt und gelagert werden. Bei der Bestimmung der Losgröße ist zwischen dem Aufwand für das Rüsten, Produktion und den Lagerhaltungskosten abzuwägen. Durch eine Losbildung steigen die Bestände in der Fertigung, was zu höheren Durchlaufzeiten und geringerer Flexibilität führt. Die erste Arbeit auf diesem Gebiet, die Produktionskosten und Lagerhaltungskosten gegenüberstellt, stammt von Harris (1913). Eine umfangreiche Darstellung von Ansätzen zur Bestimmung optimaler Losgrößen wird von Derstroff (1995) gegeben. Einen bedientheoretisch basierten Ansatz zur Losgrößenharmonisierung stellt Greiling (1998) vor.

Ist eine Losbildung nicht möglich oder nicht sinnvoll, beispielsweise bei ausgeprägten *make-to-order* Branchen wie der Investitionsgüterindustrie oder bei Umschlagprozessen, müssen reihenfolgeabhängige Rüstzeiten in die Betrachtung integriert werden.

[1] Optimierungsprobleme werden nach ihrer Komplexität unterteilt in die Klassen P und NP. Die Klassen entsprechen dem Aufwand zur Lösung des Problems abhängig von der Problemgröße. Ist die Größe eines Optimierungs-Problem durch k gegeben und kann das Problem in $f(k) = k^n$ Zeiteinheiten gelöst werden, spricht man von polynomialem Rechenaufwand, also der Klasse P. Kann das Problem nur mit exponentiellem Rechenaufwand, also $f(k) = m^k$ gelöst werden, wird das Problem der Klasse NP zugeordnet, die *nicht* in polynomialem Aufwand gelöst werden können.

Von Günther und Tempelmeier (2000) wird für eine gegebene Konfiguration eines Fertigungssystems die Kapazität der Puffer ermittelt. Vorhandene, kapazitativ begrenzte Eingangspuffer werden von Hall, Posner und Potts (1997) untersucht. Aufträge, deren Material nicht im Eingangspuffer gepuffert werden kann oder deren Fertigstellungszeitpunkte nicht eingehalten werden können, gehen verloren, es entstehen Opportunitätskosten. Hall, Posner und Potts (1997) entwickeln ein Modell mit dem Ziel, die Anzahl der verlorenen Aufträge zu minimieren. Sie untersuchen jedoch keine mehrstufigen Systeme. In einem späteren Artikel (Hall, Posner und Potts (1998)) untersuchen sie die Auswirkungen von Ausgangspuffern, auch hier wiederum nur für den einstufigen Fall. Von Khosla (1995) werden die Effekte untersucht, wenn Aufträge bei einem zweistufigen Flowshop-Problem mit mehreren Ressourcen auf der ersten Stufe um den Puffer konkurrieren.

Die Komplexität der Scheduling-Probleme hat zu der Entwicklung von zahlreichen *Prioritätsregeln* geführt, die nicht das Optimum suchen, sondern eine schnelle, evtl. suboptimale Entscheidung ermöglichen. Diese Regeln werden zu der Klasse der Heuristiken gerechnet. Hierbei wird definiert, welcher Auftrag bei Freiwerden einer Ressource als nächster zu bearbeiten ist. Diese Entscheidung kann *lokal*, d.h. unter Berücksichtigung der Warteschlange vor einer Maschine, oder *global* unter Einbeziehung zusätzlicher Informationen über das zu steuernde System getroffen werden. Wird eine Regel unabhängig von dem aktuellen Systemzustand angewendet, spricht man von *statischen*, bei systemzustandsabhängigen von *dynamischen* Regeln. Eine *elementare* Regel berücksichtigt nur eine Zielgröße, wohingegen *zusammengesetzte* Regeln mehrere Zielgrößen mittels Gewichtung miteinander verknüpfen. Für spezielle Probleme, insbesondere für Ein- und Zwei-Maschinen-Probleme, kann gezeigt werden, daß die Anwendung von Regeln zu einem optimalen Belegungsplan führt. In der Praxis sind solche einfachen Systeme leider selten anzutreffen.

Beispiele für Prioritätsregeln sind nach Panwalkar und Iskander (1977), die 113 unterschiedliche Regeln vergleichen, und Pinedo (1995):

- SIRO (service in random order): Diese Regel wird der Praxis häufig verwendet, der nächste Auftrag wird zufällig ausgewählt.
- FCFS (first come, first served), LCFS (last come, first served): Der Auftrag, der den Warteraum zuerst (zuletzt) betritt, wird als erster bearbeitet.
- EDD (earliest due date): Der Auftrag mit dem frühesten Fertigstellungszeitpunkt wird als nächster ausgewählt.
- MS (minimum slack): Der Auftrag, bei dem die Differenz zwischen dem gewünschten Fertigstellungszeitpunkt und den verbleibenden Bearbeitungs-

zeiten, der sog. *slack* oder Schlupf, am kleinsten ist, wird als nächster eingeplant. MS ist ein Beispiel für eine globale, dynamische Regel.

- ATCS (apparent tardiness cost with setup): Eine zusammengesetzte Regel, die den Eintrittszeitpunkt des Auftrages in das System und reihenfolgeabhängige Rüstzeiten berücksichtigt. Die zugrunde liegenden Regeln werden über Gewichtungsfaktoren skaliert.

In einem praktischen Umfeld entstehen solche zusammengesetzten Regeln durch Änderungen des Ressourcenparks, der Auftragsstruktur etc. Werden die Regeln hierbei nicht systematisch an die neuen Anforderungen angepaßt und überprüft, kann ein solches System ein deutlich suboptimales Verhalten aufweisen. Generell gilt, je komplexer ein System wird und je größer die stochastischen Einflüsse sind, umso einfacher sollten die verwendeten Regeln sein.

3.2. Constraint Satisfaction Ansätze

In Kapitel 5.1 werden die in dieser Arbeit verwendeten Constraint Satisfaction Ansätze näher beleuchtet, hier soll zur Verständnis eine sehr knappe Einführung gegeben werden. Ein Constraint Satisfaction Problem (CSP) besteht aus einer Menge von Variablen, deren Wertebereiche und einer Menge von Constraints, die auf den Variablen definiert sind. Das Ziel ist es, jeder Variable einen Wert zuzuordnen, so daß alle Constraints erfüllt sind. Ein einfacher Algorithmus zur Bestimmung einer Lösung ist der sog. *Backtracking*-Algorithmus. Hier werden die Variablen nach einer vorgegebenen Heuristik ausgewählt, und es wird geprüft, ob die (Partial-)Lösung noch gültig ist. Falls ja, wird die nächste Variable ausgewählt, falls nein wird die letzte Auswahl rückgängig gemacht und es wird eine neue Variable ausgewählt. Für Optimierungsprobleme wird das CSP über die Definition einer Zielfunktion in ein Constraint Optimization Problem (COP) überführt. Hier soll keine gültige, sondern die optimale Lösung gefunden werden. Einen aktuellen Überblick über Algorithmen zur Lösung von Constraint Satisfaction Problemen und Anwendungen geben Brailsford, Potts und Smith (1999).

In der Dissertation von Fox (1983) werden Constraint Satisfaction Ansätze zur Lösung von komplexen Scheduling Problemen angewendet. Das Ergebnis der Arbeit ist das Scheduling System ISIS II, das die Möglichkeit bietet, hierarchische Constraints zu modellieren, Constraints während des Suchprozesses zu relaxieren bzw. zu verstärken und schlechte Pläne zu identifizieren. Er untersucht *Konflikte*, bzw. *Interaktion* zwischen den Constraints, die *Wichtigkeit* der Constraints

und die *Erzeugung* von Constraints, die a-priori oder dynamisch bei einem sich ändernden Umfeld geschehen kann.

Wenn der Suchprozeß in einer Sackgasse mündet oder bei Verwenden der Baumsuche ein weiteres Untersuchen des aktuellen Astes nur zu einer schlechteren Lösung führen würde, ist ein Rückschreiten, der sogenannte *backtrack*-Schritt nötig. Durch Anwendung von statischen Verfahren zur Auswahl von Variablen und Werten kann es zum sogenannten *trashing* kommen, bei dem das Suchverfahren sehr oft zu dem gleichen schlechten Ast zurückkehrt. Unterschiedliche backtrack-Verfahren werden von Sadeh, Sycara und Xiong (1995a) und Sadeh, Sycara und Xiong (1995b) entwickelt, um den Suchprozeß gezielt von schlechten Lösungen wegzuleiten.

Eine Bedingung, die eine Suche ohne backtrack-Schritte garantiert, stellt Freuder (1982) vor. Sie basiert auf strenger k-Konsistenz und der Berechnung von minimalen Wegen in dem Constraint-Graphen. Der Autor gibt jedoch an, das der Rechenaufwand selbst bei kleinen Problemen sehr hoch ist.

In seiner Dissertation legt Sadeh (1991) den Grundstein zu dem Scheduling-System Micro-Boss, das auf einem sog. *micro-opportunistic* Ansatz basiert. Dieser Ansatz fußt auf der Annahme, daß der Engpaß eines Systems den Durchsatz bestimmt, daher sollten alle Ressourcen auf diesen Engpaß ausgerichtet werden. Dieser Ansatz ist deutlich detaillierter als die verbale Umschreibung des *drum-buffer-rope*-Prinzips von Goldratt und Cox (1998). Die Belegung des Engpaß sollte möglichst optimal sein. Das ist durch die Reduktion der Problemgröße auf eine Ressource auch möglich. Mit der optimalen Einplanung kann während der Aufstellung des Ablaufplanes ein neuer Engpaß entstehen, somit muß das Verfahren wiederholt werden. Sadeh entwickelt ein constraint-basiertes System, das die Belegung der Ressourcen während des Planungsprozesses überwacht, und somit auf die dynamische Änderung von Engpässen reagieren kann. Er entwirft ein *look-ahead* Verfahren, das die Entwicklung der Engpässe, d.h. die zu erwartende Belegung während der Planung, überwacht. Weiterhin zeigt er eine Abhängigkeit zwischen dem Detailliertheitsgrad von Unterproblemen und der Effizienz von Backtracking-Verfahren. Er entwickelt Heuristiken zur Auswahl von Variablen und Werten, die die Anzahl und den Aufwand der backtrack-Schritte reduzieren, u.a. in Sadeh und Fox (1996). Sadeh dokumentiert für seinen Ansatz bessere Laufzeiten im Vergleich zu speziell entwickelten Heuristiken und generischen CSP Heuristiken.

Die Idee des Kapazitäts-Bedarfs von Sadeh (1991) (der Autor spricht von *demand profile*) wird von Breitinger und Lock (1994) aufgegriffen. Sie entwickeln *semi-dynamische* Heuristiken zur Auswahl von Variablen. Hier werden die Bedarfs-Profile nicht nach jeder Zuweisung neu berechnet, sondern am Anfang wird eine Zuweisung getroffen, die während der Suche in vorgegebenen Schritten angepaßt

wird. Außerdem entwickeln sie eine Heuristik, um die Konsistenz bei globalen Constraints sicherzustellen. Das Ziel ist die Minimierung der Durchlaufzeit bei einem 10×10-Problem; es werden keine alternativen Ressourcen oder reihenfolgeabhängige Übergänge berücksichtigt.

Von Beck (1999) werden Verfahren entwickelt, die die Struktur des Suchraumes während der Suche analysieren und somit den Prozeß der Lösungsfindung verkürzen können. Er gibt einen Überblick, sowie einen umfassenden Laufzeitvergleich über Verfahren zur Sortierung von Variablen und Werten.

Die Einschränkung der Wertebereiche ist Gegenstand der Arbeiten von Caseau und Laburthe (1994) und Goltz (1995). Hier werden Eigenschaften der einzuplanenden Aktivitäten ausgenutzt, um ungültige Lösungen a-priori durch Einschränkung des Wertebereiches und damit des Suchraumes zu vermeiden.

In seiner sehr interessanten Dissertation beschreibt Nuijten (1994) die Lösung des sog. *TRCSP* (Time and Ressource Constraint Scheduling Problem) als Constraint Satisfaction Problem. Der Hauptteil seiner Arbeit besteht in der Entwicklung von Algorithmen zur Konsistenz-Prüfung und damit der Möglichkeit, den Suchraum einzuschränken. Er unterscheidet hierbei zwischen *notwendigen* Constraints, die die Einhaltung der Kapazitätsbeschränkung der Ressourcen (oder Mengen von Ressourcen) garantieren, und *zusätzlichen* (temporalen und ressourcenabhängigen) Constraints. Die entwickelten Algorithmen werden auf theoretische und zwei praktische Probleme angewendet. Er entwickelt Konsistenzalgorithmen bei Verwendung von alternativen Ressourcen, macht jedoch keine Aussagen über reihenfolgeabhängige Übergänge. Zur Auswahl von Variablen und Werten wird der frühest mögliche Fertigstellungszeitpunkt verwendet. Aus den Operationen, deren möglicher Starttermin früher liegt, wird eine zufällig ausgewählt. Ebenso wird bei der Auswahl einer Ressource aus einer Menge von alternativ möglichen Ressourcen vorgegangen. Auch das verwendete Verfahren, um aus ungültigen Zuständen des Suchbaumes herauszukommen, ist sehr einfach. Nuijten verwendet das einfache chronological backtracking, falls hier keine Lösung gefunden wird, wird das Verfahren bei der Wurzel des Suchbaumes neu gestartet. Mit einer Zufallssuche soll sichergestellt werden, daß nicht der gleiche Weg durch den Suchraum beschritten wird. Trotz dieser einfachen Suchverfahren sind die Ergebnisse sehr gut, was auf die effiziente Einschränkung des Suchraumes a-priori durch die Konsistenz-Prüfung zurückzuführen ist.

Die Entscheidung, wann welche Operation ausgeführt wird, sollte nach Cheng und Smith (1997) möglichst spät im Suchprozeß getroffen werden. Sie identifizieren aus den *explizit* angegebenen Constraints *implizite* Reihenfolge-Constraints. Das $Ordering(i, j)$ definiert die Reihenfolge zwischen den beiden Operationen i und j. Das Ziel ist es, möglichst viele Orderings zu finden, und so den Aufwand

für die Suche nach Lösungen zu verringern. Kann aus den Orderings eine Kette erzeugt werden, ist die Bestimmung der Lösung trivial. Weiterhin führen sie das Problem der Minimierung der Durchlaufzeit auf eine Menge von *deadline-scheduling* Problemen zurück. Sie erwähnen keine alternativen Ressourcen oder Reihenfolgeabhängigkeiten.

Baptiste, le Pape und Nuijten (1995) kombinieren einen constraint-basierten Approximationsalgorithmus und einen constraint-basierten Optimierungsalgorithmus, um Job-Shop-Probleme zu lösen. Trotz des allgemein formulierten Ansatzes konnte eine bislang ungelöste Variante eines Scheduling-Problems gelöst werden.

In dem Artikel von Wallace (1993) werden im wesentlichen bekannte Verfahren und Ansätze zur Verwendung von Constraint-Ansätzen im Scheduling beschrieben. Interessant ist die Übertragung des Konzeptes der *nogoods* auf kombinatorische Optimierungsprobleme. Die nogoods werden erstmals von Stallman und Sussman (1977) beschrieben, siehe auch Shanahan und Southwick (1989). Die Idee ist die Vermeidung der Suche in symmetrischen Ästen des Suchbaumes. Sind A, B, C einzuplanende Operationen, ist es für die Gesamtdurchlaufzeit irrelevant, ob die Reihenfolge $(A \rightarrow B \rightarrow C), (C \rightarrow B \rightarrow A)$ oder jede andere Permutation ist. Führt eine Reihenfolge zu einer ungültigen Lösung, handelt es sich um ein nogood, und alle Permutationen dieser Zuweisung müssen nicht mehr untersucht werden. Die Anwendung bei reihenfolgeabhängigen Übergangszeiten ist problematisch, wenn Asymmetrie unterstellt wird, d.h. $A \rightarrow B \neq B \rightarrow A$ gilt.

Von Le Page und Barras (1997) wurde ein constraint-basierter Ansatz entwickelt, um mobile autonome Roboter in einem intermodalen Terminal zu steuern. Hierbei wird für jeden umzuschlagenden Container der Roboter und ein abzufahrender Weg ausgewählt. Zur Kollisionsvermeidung der Trajektorien (Wege) der Roboter wird bei Vergabe eines Auftrages ein CSP modelliert. Als Ergebnis liegt die Trajektorie des Roboter vor.

Die Beachtung von begrenzten Pufferressourcen wurde von Simonis und Cornelissens (1995) und hierauf aufbauend in der Dissertation von Beck (1999) behandelt. Sie untersuchen die Modellierung von produzierenden und konsumierenden Ressourcen, die durch einen Puffer voneinander entkoppelt sind. Beck (1999) bestimmt obere und untere Schranken und leitet hieraus zusätzliche Reihenfolgebedingungen ab.

3.3. Mehrstufige Kommissioniersysteme

Kommissionieren hat das Ziel, aus einer Gesamtmenge von Gütern (Sortiment) Teilmengen auf Grund von Anforderungen (Aufträge) zusammenzustellen, VDI 3590 (1994).

Kommissioniersysteme werden nach VDI 3590 (1994) unterteilt nach der

- *Bereitstellung* der Artikel, die statisch oder dynamisch sein kann,
- *Fortbewegung* des Kommissionierers, die ein- oder zweidimensional erfolgen kann,
- *Entnahme* der Artikel, die manuell oder automatisch durchgeführt wird,
- *Abgabe* der Artikel an einem zentralen oder dezentralen Ort.

Ein Überblick über Modelle zur Optimierung des Durchsatzes, der benötigten Lagerkapazität und der Planung von Distributionszentren wird von Cormier und Gunn (1992) gegeben.

Der größte Teil der Arbeiten zur Optimierung von Kommissionierbereichen fokussiert auf Systemen mit statischer Bereitstellung. Interessant sind Ansätze für den operativen Betrieb, d.h. die Bestimmung von wegoptimalen Rundgängen unter Berücksichtigung von Gewichts-, Laderaum sowie zeitlichen Restriktionen. Hier basieren die meisten Ansätze auf der Erweiterung des klassischen Travelling Salesman Problems. Der erste Schritt besteht in der Clusterung, d.h. Zuordnung der Aufträge oder Pickpositionen zu einem Rundgang (Trip). In der zweiten Stufe wird für die Rundgänge die Route, also die wegoptimale Reihenfolge der Fächer, bestimmt. Der Begriff der Batch-Bildung ist nicht eindeutig definiert und soll daher als das Zusammenfassen von Aufträgen verstanden werden. Ein Batch kann durchaus Aufträge aus unterschiedlichen Touren oder Lagerzonen umfassen.

Der erste Ansatz auf dem Gebiet wurde von Ratliff und Rosenthal (1983) verfolgt. Hier werden einfache Kommissionierstrukturen in einen Graphen transformiert, um exakte und heuristische Verfahren zur Lösung eines modifizierten Travelling Salesman Problems anzuwenden. Für den Fall, daß der Kommissionierer zum Entnehmen von Artikeln auf die gegenüberliegenden Seiten des Ganges die Position wechseln muß, wird von Goetschalckx und Ratliff (1988) ein Algorithmus entwickelt. Auch hier werden das Layout und die anzulaufenden Positionen in einen Graph transformiert, auf den die Algorithmen angewendet werden. Von Hall (1993) werden Approximationen der Weglänge in manuell betriebenen Kommissionierbereichen entwickelt. Eine Generalisierung für Layouts mit Abkürzungen

wird von Roodbergen und Koster (1998) vorgestellt, auch dieser Ansatz basiert auf einem TSP. Sie führen u.a. eine Sensitivitätsanalyse durch, um zu ermitteln, welchen Einfluß die Nutzung von Abkürzungen hat. Der Fall, daß ein Artikel in unterschiedlichen Fächern gelagert werden kann, wird von Daniels, Rummel und Schantz (1998) untersucht. Ruben und Jacobs (1999) untersuchen die Auswirkung von Auftrags-Spitzen in einem manuellen Kommissionierbereich. Da es bei dem kurzfristigen Einsatz von mehr Mitarbeitern zu Verstopfungen kommen kann, schlagen sie eine Heuristik vor, die die Fächer abhängig von der Zugriffshäufigkeit belegt. Allen Autoren ist gemeinsam, daß ein abgeschlossener Bereich untersucht wird. Die nachgelagerten, über Puffer entkoppelten Pack- bzw. Versandbereiche und die damit entstehenden Synchronisationsprobleme werden nicht mit einbezogen.

Für den Fall der zweidimensionalen Bewegung, beispielsweise in einem automatischen Kleinteile-Lager (AKL) wird von Elsayed, Lee, Kim und Scherer (1993) ein Ansatz vorgeschlagen, der die Abweichung von gegebenen Fertigstellungszeitpunkten der Aufträge bestraft. Die Reihenfolge der Aufträge wird von Prioritäten bestimmt, die aus der gewichteten Summe der Zeit, die ein Auftrag zu spät oder zu früh ausgelagert wird, ermittelt werden. Weiter wird die Zusammenstellung der Batches und die Freigabe der Aufträge untersucht. Ein Ansatz zur Batchbildung, der auch auf Ähnlichkeiten von Aufträgen basiert, wird von Rosenwein (1996) beschrieben, allerdings können die Artikel hier nicht umsortiert werden. Unterschiedliche Metriken zur Batch-Bildung werden untersucht. Ein stochastisches Optimierungsmodell zur Maximierung des Durchsatzes unter systembedingten Restriktionen wird von Azadivar (1986) beschrieben. Auch hier wird eine Regalgasse isoliert betrachtet. Tabu Search und ereignisdiskrete Simulation wurde von Gutenschwager, Spieckermann und Voß (1998) verwendet, um die Reihenfolge der zu bearbeitenden Aufträge in einem AKL zu bestimmen.

Bei der dynamischen Bereitstellung der Artikel wird vor allem im anglo-amerikanischen Raum der Spezialfall untersucht, daß sich der Kommissionierer am Ende des Regalganges befindet, daß sog. „end-of-the-aisle order-picking“. Diese Anordnung ist häufig bei automatischen Kleinteilelagern (AKL) zu finden. Hier werden Modelle zur Zuweisung von Artikeln zu Fächern und Anfahr-Reihenfolgen der Fächer unter dem Ziel des maximalen Durchsatzes untersucht. Stellvertretend seien hier die Arbeiten von Bozer und White (1984) und Goetschalckx und Ratliff (1990) genannt. In großen Distributionszentren sind die Kommissionierer durch Förderstrecken und Puffer von dem Hochregallager entkoppelt, jede Gasse kann (theoretisch) jeden Kommissionierplatz versorgen. Bei diesen Systemen rücken die genannten Ansätze zur Durchsatzmaximierung der einzelnen Regale in den Hintergrund, hier sind integrierte Betrachtungen auch über die nachgelagerten

Zonen interessant.

Von Alicke und ten Hompel (1999) wird ein Objekt/Ebenenmodell der Lagerverwaltung entwickelt, das sich an die objektorientierte Modellierungstechnik (OMT) von Rumbaugh, Blaha und Premerlani (1993) anlehnt. Hier wird ein Referenzsystem geschaffen, das aus (atomaren) Elementen und hieraus aggregierten Objekten und Modulen besteht. Diese Modellierung macht eine standardisierte Anwendung von Optimierungsverfahren möglich.

Bei komplexen Kommissioniersystemen sollte der Engpaß die Steuerung des Systems übernehmen. In dem Verfahren KOMAX von Alicke und Arnold (1997b) und Alicke und Arnold (1997a) wird die Methode der ereignisdiskreten Simulation mit linearer Optimierung kombiniert, um die Belastung des Fördersystems bei einem mehrstufigen Kommissioniersystem zu minimieren. Die Simulation ermöglicht die Parametrierung des linearen Programms und dient zur anschließenden Überprüfung der dynamischen Einflüsse. In Alicke, Arnold und Schweitzer (1998) wird der Ansatz um die Reihenfolgebildung an der ersten Stufe erweitert. Das Modell zur Optimierung der Reihenfolge wird als Constraint Optimization Problem formuliert.

Die Berücksichtigung von Personal bei der kurzfristigen Steuerung von mehrstufigen Kommissioniersystemen wurde in der Literatur noch nicht behandelt. Ansätze zur Produktionsplanung bei flexibler Personalkapazität werden in Günther (1989) dargestellt. Die beschriebenen Flexibilisierungsansätze der Arbeitszeit können auf mehrstufige Kommissioniersysteme übertragen werden, allerdings ist der Planungshorizont kleiner.

Eine Anleitung für die Planung und sehr deutliche Darstellung der Zusammenhänge und Materialströme in komplexen Kommissioniersystemen wird von Yoon und Sharp (1996) beschrieben, auf operative Fragestellungen wird leider nicht eingegangen.

3.4. Hoist Scheduling

Probleme, die beim sog. *Hoist Scheduling* auftreten, sind sehr ähnlich zu dem in Kapitel 7 untersuchten Container-Terminal. Während der Galvanisierung, beispielsweise für die Herstellung von Leiterplatten, werden die Produkte in einer vorgegebenen Reihenfolge in Tanks behandelt (galvanisieren, spülen, etc.). Um die vorgegebene Qualität einzuhalten, müssen die Aufenthaltszeiten für jeden Prozeßschritt in einem gegebenen Zeitfenster liegen, d.h. es sind minimal und maximal erlaubte Behandlungszeiten gegeben. In automatischen Produktionsstraßen wird der Transport zwischen den einzelnen Schritten mit automatischen Kränen reali-

siert, die auf Schienen oberhalb der Tanks operieren. Eine einfache serielle Bearbeitungsreihenfolge benutzt jeden Tank genau einmal. Aus Kostengründen werden jedoch bestimmte Tanks mehrfach verwendet (z.B. bei Spül- oder Trockenbädern). Weiterhin wird die Belade- und Entladestation oft zusammengelegt, siehe Abbildung 3.1.

Das *Hoist Scheduling Problem* besteht darin, für die eingesetzten Kräne eine Bearbeitungsreihenfolge zu bestimmen, die Restriktionen wie gegebene Zeitfenster erfüllt und Zielkriterien wie Durchlaufzeiten minimiert. Wird immer das gleiche Produkt oder der gleiche Mix von Produkten bearbeitet, kann die ermittelte Reihenfolge immer wiederholt werden, man spricht hier von einem Zyklus und daher auch von *Cyclic Hoist Scheduling*.

Die in der Literatur untersuchten Probleme können unterschieden werden nach

- der Anzahl der eingesetzten Kräne in *single-hoist* bzw. *multi-hoist*,
- der Behandlungszeit in *fixed*, wenn ein diskreter Wert gegeben ist, oder *bounded* wenn ein Zeitfenster definiert ist,
- den Einsatzbereichen der Kräne, die *disjunkt* oder *überlappend* sein können.

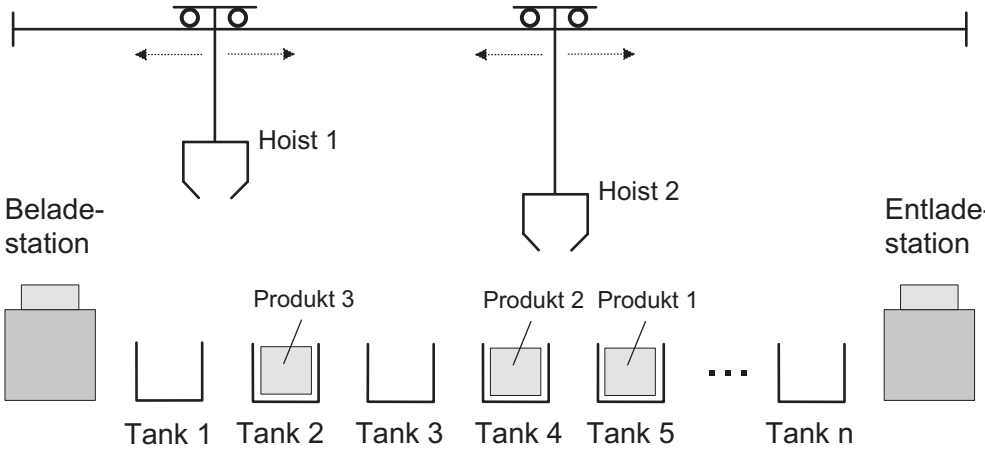


Abbildung 3.1.: Darstellung eines Cyclic Multi Hoist Scheduling Problems mit überlappenden Einsatzbereichen.

Den Unterschied zu klassischen flow-shop oder job-shop Problemen sehen Varnier, Bachelu und Baptiste (1997) in den folgenden Punkten:

- Die Behandlungszeit in den einzelnen Tanks liegt nicht als diskreter Wert vor, sondern wird durch das zugrunde liegende Verfahren lediglich als Mini-

mum und Maximum der Behandlungszeit vorgegeben. Die Zeit ergibt sich a-posteriori aus der realisierten Reihenfolge.

- Die Übergangszeiten zwischen den einzelnen Operationen können nicht vernachlässigt werden.
- Pufferung außerhalb der Tanks (und damit Wartezeit) ist nicht möglich.

Single Hoist-Probleme werden von zahlreichen Autoren untersucht Ng (1996), Hanen (1994) und Chen, Chengbin und Proth (1998). Yih (1994) nutzt mögliche Konflikte durch die Stationen, die Verfügbarkeit und den Aufenthaltsort des Kranes in seinem Algorithmus, um für ein Problem mit Zeitfenstern eine Lösung mit minimaler Durchlaufzeit zu bestimmen. Von Phillips und Unger (1976) wurde ein gemischt-ganzzahliger Ansatz entwickelt, um ein Single Hoist Problem zu lösen, bei dem kein Tank doppelt benutzt wird und die Reihenfolge der Bearbeitungsschritte dem Materialfluß entspricht. Der Kran fährt also in eine Richtung immer voll oder leer und in die andere ausschließlich leer. Von Lei und Wang (1991) wurde gezeigt, daß dieses (einfache) Problem bereits NP-vollständig ist.

Im Fall des Multi Hoist Problems kommt zu der Bestimmung einer optimalen Reihenfolge noch die Zuordnung der einzelnen Transporte zu den verwendeten Kränen oder die Bestimmung von Kranbereichen hinzu. Dieses Problem des simultanen *scheduling* und *assignment* wird von Varnier, Bachelu und Baptiste (1997) als „very combinatorial problem“ klassifiziert. Sie merken an, daß hier noch keine Lösung existiert, und entwickeln einen zweistufigen Ansatz. Im ersten Schritt werden die Transportaufträge so zugewiesen, daß die Fahr- und Handlingszeit der Kräne möglichst gleichmäßig verteilt ist. Da jeder Transportauftrag theoretisch jedem Kran zugeordnet werden kann, sind Konfliktsituationen durch Kollision möglich. Durch Einführen von Reihenfolgebedingungen zwischen zwei Kranbewegungen, die sich überlagern würden, werden Kollisionen vermieden. Die Autoren benutzen Constraint Logic Programming, um die Reihenfolgeplanung des zweiten Schrittes durchzuführen.

Lei und Wang (1991) entwickeln für ein Problem mit zwei Kränen und disjunkten Einsatzbereichen eine Heuristik, die auf einem Partitionierungsansatz aufbaut. Die Idee hierbei ist die Aufteilung in zwei Bereiche, für die jeweils ein Single Hoist Problem formuliert und optimal gelöst wird. Die beiden Lösungen werden nun mit Hilfe einer Heuristik zu einer gültigen Gesamtlösung kombiniert. Experimente zeigen, daß in einer akzeptablen Laufzeit gute Ergebnisse erzielt werden.

Von Rodosek und Wallace (1998) werden gemischt-ganzzahlige Programmierung, Constraint Logic Programming und eine Kombination aus beiden Verfahren (die als hybrider Ansatz bezeichnet wird) auf unterschiedliche Problemstellungen im

Bereich des Hoist Scheduling angewendet. Die Autoren berichten von dem in ihren Augen erstaunlichem Ergebnis, daß der hybride Ansatz die kürzeste Laufzeit liefert.

3.5. Das Spannungsfeld zwischen theoretischen Annahmen und praktischen Anforderungen

Die Ziele der Forschung und der Praxis sind oft unterschiedlich. So will die Praxis eine schnelle, möglichst gute Lösung, wohingegen die Theoretiker auf der Suche nach dem Optimum sind. Aufgrund der Komplexität von praktischen Problemen ist die Berechnung eines Optimums nur in nahezu unendlicher Zeit möglich (Hopp und Spearman (1996) geben ein schönes Beispiel), daher sind also gute praktisch anwendbare Lösungen gefragt. Hillier und Lieberman (1995) sprichen von „Satisfizing“, einer Kombination aus „Satisfactory“ und „Optimizing“. Es gilt also auch die Zeit, die für das Optimierungsprojekt und die Berechnung der Lösung benötigt wird, in die Zielfunktion mit einzugliedern (nur hypothetisch gesehen). In der universitären Forschung kann dieser (Zeit-) Term mit einem geringeren Gewicht versehen werden, aber er sollte nicht vergessen werden.

Von Pinedo (1995) wird die Kritik an theoretischen Verfahren in den folgenden Punkten konkretisiert:

- Bei theoretischen Ansätzen wird nicht rollierend geplant, d.h. n Aufträge werden eingeplant und nachdem alle abgearbeitet sind, können die nächsten n Aufträge verplant werden. Es wird davon ausgegangen, daß das System zu Beginn der Untersuchung *leer* ist. In der Regel sind allerdings bereits eingeplante, aber noch nicht fertiggestellte Aufträge im System vorhanden. Diese Aufträge müssen berücksichtigt werden. Durch stochastische Einflüsse wie Maschinenausfälle, Schwankungen und Änderungen der Nachfrage wird eine regelmäßige Neuplanung in Form einer rollierenden Planung oder Online-Optimierung nötig.

- Mathematische Modelle können *weiche* Restriktionen nur über Strafterme in der Zielfunktion berücksichtigen. So ist die Kapazität der Ressourcen beispielsweise nicht fix, sondern kann durch eine zusätzliche Schicht vergrößert werden.

- Die Bestrafungsfunktionen der Praxis sind nicht-linear oder stückweise linear. Theoretische Arbeiten konzentrieren sich oft auf ein Zielkriterium, in der

Praxis wird versucht, eine Kombination von Zielgrößen zu berücksichtigen. Hierbei spricht man von *mehrkriteriellen* Problemen.

- Die Gewichtung der Zielkriterien kann dynamisch und systemabhängig sein, z.B. bei einer geringen Auslastung die Minimierung der gewichteten Verspätung, bei einer hohen Auslastung die Minimierung der Summe der Rüstzeiten an den Engpaß-Ressourcen.

Derrick und Pegman (1998) kritisieren weiterhin, daß die Anwendung eines Optimierungsmodells auch das Umfeld berücksichtigen sollte. Die Autoren merken an, ein Planer „can't stick to nice clean academic problems and models - real life tends to be much messier", wobei angefügt wird, daß diese Aussage nicht die Modelle kritisieren soll, sondern als eine Aufforderung zu sehen ist, den Horizont der Modelle zu erweitern und damit anwendbar zu machen.

3.6. Zusammenfassung

Auf dem Gebiet des Scheduling existieren zahlreiche Ansätze zur Lösung theoretischer Probleme. Die Notation $n \times m$ wurde eingeführt, um die Probleme zu klassifizieren. Durch Weiterentwicklung der Rechnerleistung und neue Verfahren wurde das berühmt-berüchtigte 10×10 - Problem vor einigen Jahren optimal gelöst. Die Weiterentwicklung der Verfahren bringt aus theoretischer Sicht nur wenig Verbesserung oder Erfolg. Nun sollten praktische Probleme gelöst werden. Theoretische Verfahren treffen eine Vielzahl von Annahmen, daher ist die Rechenzeit bei der Anwendung auf praktische Problemstellungen oft enttäuschend.

Ein wichtiger Bestandteil der Lösung von praktischen Problemen ist die Modellierung geworden. Verfahren aus dem Bereich der künstlichen Intelligenz (AI) sind näher an der Beschreibung von Optimierungsproblemen orientiert als die sehr speziellen Algorithmen des Operations Research (OR). Die Grenze ist allerdings fließend. Hier wurde in den letzten Jahren der Ansatz des Constraint Programming konsequent weiterentwickelt und auf komplexe Optimierungsprobleme unter Beachtung von praxisrelevanten Nebenbedingungen angewendet.

Ähnliche Probleme wie bei mehrstufigen Umschlagsystemen sind bei Hoist Scheduling Problemen und komplexen Kommissioniersystemen anzutreffen. Auch hier werden nicht alle Aspekte der untersuchten Systeme abgedeckt, da zahlreiche Annahmen getroffen bzw. nur Teilsysteme optimiert werden.

Die Kritik (von Theoretikern) an theoretischen Ansätzen schließt das Kapitel ab.

4. Modellierung mehrstufiger Systeme

„All models are wrong, but some are useful“ (George Box)

Die Modellierung ist ein kreativer Prozeß, bei dem ein reales System in ein abstraktes Modell[1] überführt wird, um gezielt zu experimentieren. Die Erkenntnisse aus der Experimentierphase können benutzt werden, um das modellierte System kurz- oder langfristig zu verbessern, zu *optimieren*[2].

Die *Systemtheorie* beschäftigt sich nach Brockhaus (1983) „mit der Erforschung des Zusammenwirkens der durch ihre Einzelfunktionen beschriebenen Elemente eines Systems miteinander und mit der Außenwelt sowie der Beziehungen zwischen gekoppelten Systemen .... Wesentliche Aspekte der allgemeinen Systemtheorie sind die Klassifikation von Systemen und die mathematische Erfassung von Beziehungen und Gesetzen ... “.

Systeme können nach ihrer Zustandsänderung in stochastische und deterministische unterteilt werden, die Planung und Steuerung solcher Systeme kann langfristig und kurzfristig geschehen. Die hier untersuchten Systeme sind determini-

[1] „Modell: Nachbildung oder Entwurf von Gegenständen aller Art in zum Teil kleinerem Maßstab. In der Naturwissenschaft soll das Modell die wesentlichen Eigenschaften des Vorbildes ausdrücken, nebensächliche Eigenschaften außer acht lassen, um durch diese Vereinfachung zu einem übersehbaren Gedankengang oder zu einem mathematisch berechenbaren ... Gegenstand zu kommen ...“ nach Brockhaus (1983)

[2] Brockhaus (1983) definiert Optimieren als „Aufsuchen des kleinsten (**Minimierung**) oder größten (**Maximierung**) Wertes einer Funktion (**Zielfunktion**) in einem bestimmten, durch Nebenbedingungen, oft in Form von Gleichungen und Ungleichungen beschriebenen (zulässigen) Bereich.... Optimierungsaufgaben haben auch besondere Bedeutung ... für die Maximierung des Gewinns bei beschränkten Produktionsfaktoren (Kapital, Maschinen, Arbeitskraft) oder für die Minimierung der Kosten. Optimierungsaufgaben treten häufig im Zusammenhang der modellgestützten Entscheidung des **Operations Research** (die Anwendung mathematischer Methoden und Modelle zur Vorbereitung optimaler Entscheidungen) auf.“

stisch, die Planung und Steuerung betrifft nur den kurzfristigen Betrieb. Bei den praktischen Anwendungsfällen (Kapitel 6 und 7) ist die schnelle Reaktion auf unvorhergesehene Störungen wie Verspätungen, Ausfall von Ressourcen wichtig.

Stark vernetzte, komplexe Systeme können *reduktionistisch*, d.h. jedes für sich, oder *holistisch*, also im Ganzen untersucht werden. Durch eine reduktionistische Sichtweise werden Auswirkungen von lokalen Entscheidungen auf nachgelagerte und damit abhängige Elemente vernachlässigt. Das kann zu suboptimalen und ungültigen Lösungen führen. Holistische Ansätze modellieren Systeme als Ganzes, somit werden Abhängigkeiten bei komplexen vernetzten Systemen berücksichtigt. Ein Problem bei holistischen Ansätzen ist die Komplexität der Modelle und die damit verbundene Rechenzeit.

Hall (1985) vergleicht die Modellierung von Optimierungsproblemen mit der Physik. Hier ist eine sehr detaillierte Sicht schädlich und führt zu einer unnötigen Komplexität, es ist schwer, die gewonnenen Erkenntnisse zu übertragen. Vielmehr sollte das Ziel die Betrachtung auf einem aggregierten Niveau, mit anschließender, punktueller Detaillierung sein.

Ein Modell zur Lösung von Optimierungsproblemen kann ebenfalls beliebig detailliert oder aggregiert sein. Die Wahl des richtigen Aggregationsgrades ist von entscheidender Bedeutung für die Aussagekraft und Laufzeit. Ein komplexes Fördersystem läßt sich beispielsweise in Blockabschnitte unterteilen, um an jeder Verzweigung und Zusammenführung die (Bearbeitungs-)Reihenfolge der Förderelemente exakt zu modellieren. Allerdings wäre das entstehende Modell zu komplex und unhandlich, die Ergebnisse ließen sich schwer auf die Praxis übertragen, da durch kleinste stochastische Einflüsse eine Neuberechnung nötig wäre. Das Modell ist nicht *robust*. Hier bietet sich eine Aggregation des Fördersystems mit einer fundierten Durchsatzberechnung an. Auf der anderen Seite führt eine Vernachlässigung des Puffers zwischen den einzelnen Stufen auch zu unrealistischen Ergebnissen.

Die Kunst des Modellierens liegt darin, den Aggregationsgrad richtig festzulegen oder in einem hierarchischen Modell an den richtigen Stellen die wichtigen Details zu untersuchen.

Der hier verfolgte Modellierungsansatz ist systemtheoretisch basiert, d.h. es werden Elemente definiert, die miteinander in Beziehungen stehen. Es wird ein holistisches Modell des untersuchten Systems entwickelt. Die Elemente eines mehrstufigen Umschlagsystems werden in Kapitel 4.1 dargestellt, die Beziehungen zwischen den Elementen werden in Kapitel 4.2 eingeführt. Unter anderem wird eine Unterscheidung in puffernde und bearbeitende Ressourcen vorgenommen, reihenfolgeabhängige Übergangszeiten werden integriert. In Kapitel 4.3 werden mehrstufige

Systeme beschrieben, um die Grundlage für die Anwendung zu schaffen.

4.1. Elemente

Der Umschlagsvorgang einer Ladeeinheit kann als ein *Auftrag* verstanden werden, der als eine Folge von *Operationen* von unterschiedlichen *Ressourcen* ausgeführt wird, diese also für eine bestimmte Zeit in Anspruch nimmt. Aufträge können sehr unterschiedlich sein, beispielsweise

- ein *Kunden*auftrag, bei dem bestimmte Produkte nachgefragt werden, die zu einem vorgegebenen Zeitpunkt ausgeliefert werden müssen,
- ein *Fahr*auftrag, um ein Fördergut von einer Quelle zu einer Senke zu transportieren.

Definition 1 *Ein* Auftrag *(Job)* $j \in \mathcal{J}$ *besteht aus einer Menge von* n_j Operationen $O_1^j \ldots O_{n_j}^j \in \mathcal{O}$, *die auf* Ressourcen $r \in \mathcal{R}$ *ausgeführt werden.*

Beispiele hierfür sind:

- Ein Umschlags-*Auftrag* (einer Fördereinheit), der aus den *Operationen* Aufnehmen, Transportieren und Absetzen besteht und von der *Ressource* Kran ausgeführt wird.
- Ein Kommissionier-*Auftrag*, der aus den *Operationen* Scannen, Greifen, Quittieren, etc. besteht und von der Ressource Kommissionierplatz (und Kommissionierer) ausgeführt wird.
- Ein Fertigungs-*Auftrag*, der aus Bearbeitungs-*Operationen* besteht, die an *Ressourcen* wie z.B. einer Drehmaschine ausgeführt werden, etc.

Zu Beginn wird eine formale Definition des grundlegenden *M*ehrstufigen *U*mschlag *P*roblems (MUP) gegeben. Die Definition lehnt sich an Nuijten (1994) und das dort beschriebene TRCSP (Time and Resource Constraint Scheduling Problem) an[3]. In den folgenden Kapiteln werden die Teile der Definition näher erläutert.

[3] Die Bezeichnung gemäß Nuijten wäre korrekt *Time and Ressource Constraint Scheduling Problem with Sequence-Dependent Setups, Release and Due Times.* Aus Gründen der Übersichtlichkeit wird das Problem kurz als MUP bezeichnet.

Definition 2 *Die Daten, um ein MUP zu definieren, sind gegeben als n-Tupel* $SD = (\mathcal{O}, \mathcal{R}, fr, cp, sz, pt, t^{rel}, t^{due}, \succ, t^{set}, H)$. *Die Menge an auszuführenden Operationen ist* $\mathcal{O}$. *Die Menge aller Ressourcen des Systems wird mit* $\mathcal{R}$ *bezeichnet. Die Abbildung* $fr : \mathcal{O} \to \mathcal{P}(\mathcal{R})$ *definiert für jede Operation die* Menge von Ressourcen, *auf der diese Operation ausgeführt werden kann. Die* Kapazität *einer Ressource wird mit* $cp : \mathcal{R} \to \mathbb{N}^+$, *der* Bedarf *einer Operation an Ressourcen mit* $sz : \mathcal{O} \to \mathbb{N}^+$ *bezeichnet. Die* Zeit *für die Ausführung einer Operation auf einer möglichen Ressource ist gegeben durch die Abbildung* $pt : \mathcal{O} \times \mathcal{R} \to \mathbb{N}^+$. *Kann eine Operation nur auf einer Ressource ausgeführt werden, gilt* $pt : \mathcal{O} \to \mathbb{N}^+$. *Für jede Operation ist mit* $t^{rel} : \mathcal{O} \to \mathbb{N}_0^+$ *der frühest mögliche Startzeitpunkt und mit* $t^{due} : \mathcal{O} \to \mathbb{N}^+$ *der (gewünschte) Fertigstellungszeitpunkt gegeben. Durch die binäre Relation* $\succ: \mathcal{O} \times \mathcal{O}$ *wird eine Reihenfolge zwischen den Operationen festgelegt. Reihenfolgeabhängige Übergänge sind durch* $t^{set} : \mathcal{O} \times \mathcal{O} \times \mathcal{R} \to \mathbb{N}_0^+$ *gegeben. Der Untersuchungszeitraum des MUP liegt zwischen* 0 *und* H.

Die Zeiten sind als natürliche Zahlen definiert, um das Problem als ein Constraint Satisfaction Problem mit *finiten* Wertebereichen abzubilden. Diese Definition ist zulässig, da das zugrundeliegende Zeitraster beliebig fein sein kann.

Falls die Ausführung einer Operation unterbrochen und zu einem späteren Zeitpunkt wieder aufgenommen werden darf, wird das untersuchte Problem als *preemptive* klassifiziert. Im Gegensatz ist bei *non-preemptive* keine Unterbrechung erlaubt. Im folgenden werden non-preemptive Probleme behandelt.

Mit Hilfe dieser Daten läßt sich ein Belegungsplan (Schedule) σ aufstellen, der jeder Operation aus SD eine Ressource $ra : \mathcal{O} \to \mathcal{R}$, $ra(O_i^j)$ und einen Startzeitpunkt $t_s : \mathcal{O} \to N$, $t_s(O_i^j)$ zwischen 0 und H zuordnet. Ein Schedule ist gültig, wenn die in den folgenden Abschnitten vorgestellten Constraints erfüllt sind.

Aus einem gegebenen Startzeitpunkt $t_s(O_i^j)$ läßt sich der Fertigstellungszeitpunkt $t_e(O_i^j)$ einer Operation berechnen zu

$$t_e(O_i^j) = t_s(O_i^j) + pt(O_i^j, ra(O_i^j)) \tag{4.1}$$

Belegungspläne können sehr anschaulich in einem nach Henry L. Gantt benannten Gantt-Diagramm visualisiert werden. Die Operationen auf jeder Ressource werden als Funktion der Zeit dargestellt. Aus einem Gantt-Diagramm lassen sich Informationen wie die Durchlaufzeit der Aufträge, die Übergangszeit zwischen zwei Aufträgen und die Leerlaufzeit der Ressourcen ablesen. In Abbildung 4.1 ist beispielhaft ein Gantt-Diagramm für den Umschlag von zwei Containern dargestellt.

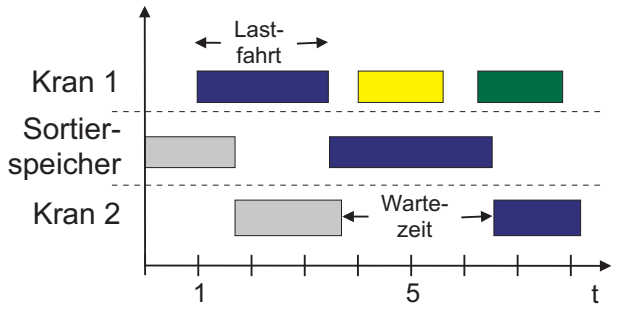


Abbildung 4.1.: Darstellung eines Gantt-Diagrammes für den Umschlag von Containern. Die Vollfahrtzeiten sind als Balken dargestellt, zusätzlich lassen sich Informationen wie Wartezeiten etc. ablesen.

4.2. Beziehungen zwischen den Elementen

Zwischen den Elementen eines Systems (Ressourcen und Operationen) können Beziehungen definiert werden, die in den folgenden Kapiteln dargestellt werden. Diese Beziehungen werden als *Constraints* bezeichnet. Hier soll der Begriff Constraint, der mit Restriktion übersetzt werden kann, im Original verwendet werden, um die Übertragung in Kapitel 5 zu vereinfachen.

4.2.1. Temporale Constraints

In einem praktischen Umfeld sind für jeden Auftrag $j \in \mathcal{J}$ im Untersuchungszeitraum $0 \ldots H$ ein (frühest möglicher) *Starttermin*, engl. *release-time* $t^{rel}(j)$ und ein gewünschter oder fest vorgegebener *Fertigstellungstermin*, engl. *due-time* $t^{due}(j)$ gegeben. In einem Distributionszentrum ist die release-time beispielsweise durch den Bestell-Eingang des Auftrages und die due-time durch die geplante Abfahrtszeit des ausliefernden Lkws gegeben. Die Abarbeitung des Auftrages, d.h. das Auslagern, Kommissionieren, Transportieren und Verpacken der entsprechenden Artikel muß in diesem Zeitraum geschehen.

Die für einen Auftrag j gegebenen Zeiten lassen sich direkt auf die in dem Auftrag enthaltenen Operationen $O_1^j \ldots O_{n_j}^j$ übertragen:

$$t^{rel}(O_i^j) = t^{rel}(j) \quad \forall\, j, i = 1 \ldots n_j \tag{4.2}$$

$$t^{due}(O_i^j) = t^{due}(j) \quad \forall\, j, i = 1 \ldots n_j \tag{4.3}$$

Für die zu ermittelnden Start- t_s und Endzeitpunkte t_e der Operationen eines Schedule σ lassen sich nun die folgenden Constraints definieren:

$$t_s(O_i^j) \geq t^{rel}(O_i^j) \quad \forall\, j, i = 1 \ldots n_j \tag{4.4}$$
$$t_s(O_i^j) + pt(O_i^j) = t_e(O_i^j) \leq t^{due}(O_i^j) \quad \forall\, j, i = 1 \ldots n_j \tag{4.5}$$

Das Zeit-Intervall, in dem eine Operation ausgeführt wird, sei $I(O_i^j) = [t_s(O_i^j), t_e(O_i^j))$[4]. Die Intervalle der Bearbeitungszeit von zwei Operationen können nach Allen (1983) die in Abbildung 4.2 visualisierten Zustände annehmen. Die entsprechenden temporalen Constraints sind:

$$\begin{aligned}
I(O_i^j) \text{ before } I(O_k^l) &\Leftrightarrow t_e(O_i^j) < t_s(O_k^l) \\
I(O_i^j) \text{ equals } I(O_k^l) &\Leftrightarrow t_s(O_i^j) = t_s(O_k^l) \wedge t_e(O_i^j) = t_e(O_k^l) \\
I(O_i^j) \text{ meets } I(O_k^l) &\Leftrightarrow t_e(O_i^j) = t_s(O_k^l) \\
I(O_i^j) \text{ overlaps } I(O_k^l) &\Leftrightarrow t_s(O_i^j) < t_s(O_k^l) \wedge t_s(O_k^l) < t_e(O_1^j) \wedge t_e(O_i^j) < t_e(O_k^l) \\
I(O_i^j) \text{ during } I(O_k^l) &\Leftrightarrow t_s(O_k^l) < t_s(O_i^j) \wedge t_e(O_i^j) < t_e(O_k^l) \\
I(O_i^j) \text{ starts } I(O_k^l) &\Leftrightarrow t_s(O_k^l) = t_s(O_i^j) \wedge t_e(O_i^j) < t_e(O_k^l) \\
I(O_i^j) \text{ ends } I(O_k^l) &\Leftrightarrow t_s(O_k^l) < t_s(O_i^j) \wedge t_e(O_i^j) = t_e(O_k^l)
\end{aligned} \tag{4.6}$$

Bis auf die Relation *equals* können die beiden Intervalle $I(O_i^j)$ und $I(O_k^l)$ vertauscht werden. Somit ergeben sich für Abbildung 4.2 insgesamt 13 mögliche Reihenfolgen.

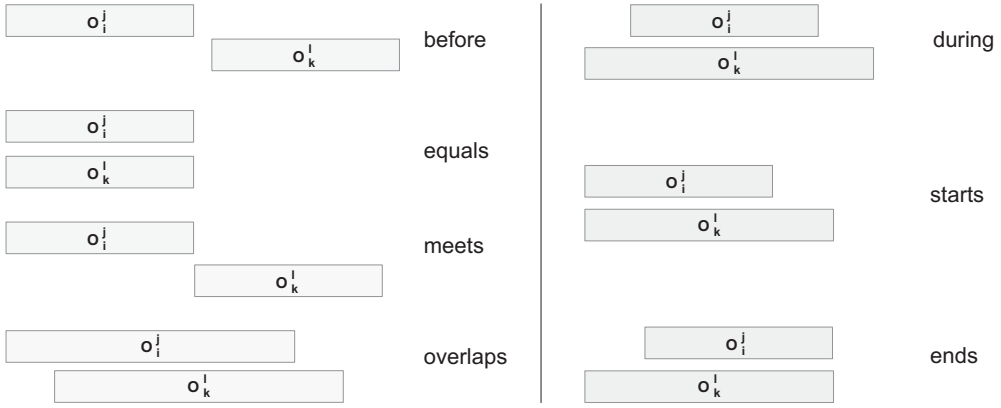


Abbildung 4.2.: Mögliche temporale Constraints zwischen den Intervallen zweier Operationen O_i^j und O_k^l.

[4] Im folgenden soll die Menge $\{a \ldots b\}$ aus Gründen der Übersichtlichkeit mit [a,b] bezeichnet werden, was als äquivalent zu [a,b) zu verstehen ist.

Ankunfts- und Abgangsprozeß

Die Elemente können das untersuchte System *seriell*, also nacheinander, oder *parallel*, d.h. batchweise betreten und verlassen. Die Ankunfts- und Abgangsprozesse können deterministisch oder stochastisch sein. Im deterministischen Fall sind die Zeiten und damit t^{rel} (Ankunft), bzw. t^{due} (Abgang) der zu bearbeitenden Elemente a-priori bekannt, im stochastischen Fall werden diese Zeiten durch Wahrscheinlichkeits-Verteilungen vorgegeben. Der stochastische Fall soll hier nicht untersucht werden, Pinedo (1995) gibt eine sehr gute Einführung in diese Problematik.

4.2.2. Reihenfolge- und Kapazitäts-Constraints

Bei einem mehrstufigen Umschlagprozeß ist eine Vielzahl von Operationen nötig, die in einer vorgegebenen Reihenfolge ausgeführt werden müssen. Bei der Herstellung von Produkten sind in der Regel Vorranggraphen und Stücklisten gegeben, bei Umschlagsystemen wie dem untersuchten Mega Hub oder dem mehrstufigen Kommissioniersystem können diese Reihenfolgebeziehungen aus dem zugrundeliegenden System abgeleitet werden. Man unterscheidet in *lineare*, *in-tree*, *out-tree* oder *generelle* Strukturen (Abbildung 4.3). Ist zwischen zwei Operationen eine Reihenfolgebeziehung definiert (O_i^j vor O_k^l), so läßt sich ein *Reihenfolge-Constraint* mit Hilfe der binären Relation $\succ$ angeben:

$$O_i^j \succ O_k^l \tag{4.7}$$

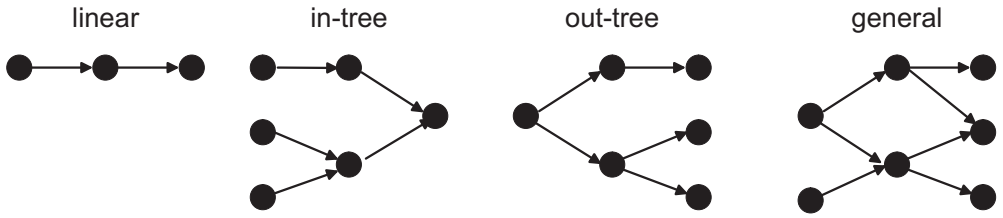


Abbildung 4.3.: Unterschiedliche Strukturen der Reihenfolgebeziehungen.

Diese Reihenfolge kann sowohl zwischen Operationen des gleichen Auftrages, als auch zwischen Operationen beliebiger Aufträge definiert werden. Die Relation (4.7) kann mit Hilfe der Startzeitpunkte und der Fertigstellungszeitpunkte der Operationen geschrieben werden als:

$$t_e(O_i^j) \leq t_s(O_k^l) \tag{4.8}$$

Als Beispiel soll die Kommissionierung eines Artikels modelliert werden. Die Operation O_1^1 stellt die Auslagerung aus dem Hochregallager und O_2^1 den Kommissionier-Vorgang dar. Die Kommissionierung darf erst begonnen werden, wenn der Artikel ausgelagert wurde, also gilt $t_e(O_1^1) \leq t_s(O_2^1)$. Hierbei wurde allerdings durch die Verwendung von $\leq$ die unrealistische Annahme getroffen, daß zwischen den einzelnen Operationen keine Übergangszeit (Transport, etc.) nötig ist. Die Beachtung von Übergangszeiten zwischen zwei aufeinanderfolgenden Operationen wird ab Seite 34 beschrieben.

Bearbeitende Ressourcen

Die Ausführung von mehreren Operationen auf einer Ressource $r \in \mathcal{R}$ führt zu Konflikten, wenn die Kapazität $cp(r)$ zu einem beliebigen Zeitpunkt überschritten wird. Für den in der Praxis oft anzutreffenden Fall, daß die Ressource eine Kapazität von $cp(r) = 1$ hat, tritt eine Überlastung auf, wenn zwei Operationen in ihrer Bearbeitungszeit überlappen. Eine gültige Lösung erhält man, wenn die beiden Operationen nacheinander ausgeführt werden, wobei das nacheinander unabhängig von ihrer Reihenfolge ist. Durch Einführen eines *Kapazitäts-Constraint* (auch als *disjunctive constraint* bezeichnet), wird dies sichergestellt:

$$\begin{aligned} &O_i^j \succ O_k^l \ \vee \ O_k^l \succ O_i^j \Leftrightarrow \\ &t_e(O_i^j) \leq t_s(O_k^l) \ \vee \ t_e(O_k^l) \leq t_s(O_i^j) \ \forall \ ra(O_k^l) = ra(O_i^j) \end{aligned} \tag{4.9}$$

Die Reihenfolge der Operationen wird durch dieses Constraint noch nicht vorgegeben, es wird jedoch sichergestellt, daß *entweder* zuerst O_i^j und danach O_k^l *oder* zuerst O_k^l und danach O_i^j ausgeführt wird. Diese Art von Constraints wird als *generisch* bezeichnet, da sie auf alle paarweisen Kombinationen von Operationen, die derselben Ressource zugewiesen wurden, angewendet werden.

Bei Ressourcen mit $cp(r) > 1$ muß sichergestellt werden, daß die Kapazität der Ressource, die sich dynamisch ändern kann[5], zu keinem Zeitpunkt überschritten wird. Für den dynamischen Fall wird der Definition der Kapazität die zeitliche Abhängigkeit hinzugefügt, $cp(r,t)$. Somit gilt für alle Operationen O_i^j, die auf einer Ressource $r \in \mathcal{R}$ zugewiesen wurden, zu jedem (diskreten) Zeitpunkt $0 \leq t \leq H$:

$$\sum_{O_i^j | ra(O_i^j)=r \wedge t_s(O_i^j) \leq t \leq t_e(O_i^j)} sz(O_i^j) \leq cp(r,t) \tag{4.10}$$

[5] Ressourcen können beispielsweise durch Schichtmodelle, Wartungszyklen etc. verfügbar oder nicht verfügbar sein. Zur Modellierung von Personal, siehe Seite 32.

Hier und in den folgenden Gleichungen wird die formal korrekte Darstellung $\{a|b\dots\}$ aus Gründen der Übersichtlichkeit durch $a|b$ ersetzt. Gleichung (4.10) ist in Abbildung 4.4 dargestellt.

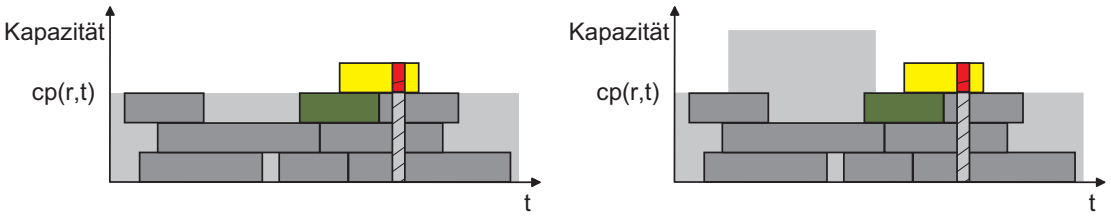


Abbildung 4.4.: Darstellung der Bedingung (4.10), links für $cp(r) = 3$ und rechts für $2 \leq cp(r,t) \leq 4$. Die schraffierte Fläche ist ein Zeitelement, in dem das Kapazitätsangebot überschritten wird.

Puffernde Ressourcen

Zur Entkopplung von Prozessen sind Puffer nötig, beispielsweise zwischen der Kommissionier- und der Packstufe eines Distributionslagers. In den Puffern kann eine Umsortierung der Produkte stattfinden, weiterhin haben Puffer eine beschränkte Kapazität $cp(r)$. Der Zu- und Abgang in einen Puffer wird von den vor- und nachgelagerten Prozessen bestimmt, wie in Abbildung 4.5 dargestellt ist. Eine Bearbeitungszeit kann aufgrund der unbestimmten Aufenthaltszeit im Puffer nicht a-priori bestimmt werden.

Die Reihenfolge von drei Operationen sei mit $O_k^j \succ O_i^j \succ O_g^j$ gegeben und O_i^j dem Puffer p zugewiesen, $ra(O_i^j) = p$. Ist kein Transport zwischen den bearbeitenden und der puffernden Ressource nötig, ergibt sich die Zeit, die der Puffer von Operation O_i^j in Anspruch genommen wird, zu $pt(O_i^j) = t_s(O_g^j) - t_e(O_k^j)$. Die Belegung des Puffers zu einem beliebigen Zeitpunkt t ergibt sich aus der Differenz der eingelagerten und ausgelagerten Produkte. Die Belegung des Puffers darf die Kapazität $cp(p)$ zu keinem Zeitpunkt überschreiten, somit muß für alle $0 \leq t \leq H$ und $ra(O_i^j) = p$ gelten:

$$\underbrace{\sum_{O_i^j|(O_k^j \succ O_i^j)\wedge(t_e(O_k^j)<t)} sz(O_i^j)}_{\substack{\text{Operation, die } sz(O_i^j) \\ \text{Einheiten eines Produktes} \\ \text{vor t in den Puffer} \\ \text{einlagert.}}} - \underbrace{\sum_{O_i^j|(O_i^j \succ O_g^j)\wedge(t_s(O_g^j)<t)} sz(O_i^j)}_{\substack{\text{Operation, die } sz(O_i^j) \\ \text{Einheiten eines Produktes} \\ \text{vor t auslagert.}}} \leq cp(p) \qquad (4.11)$$

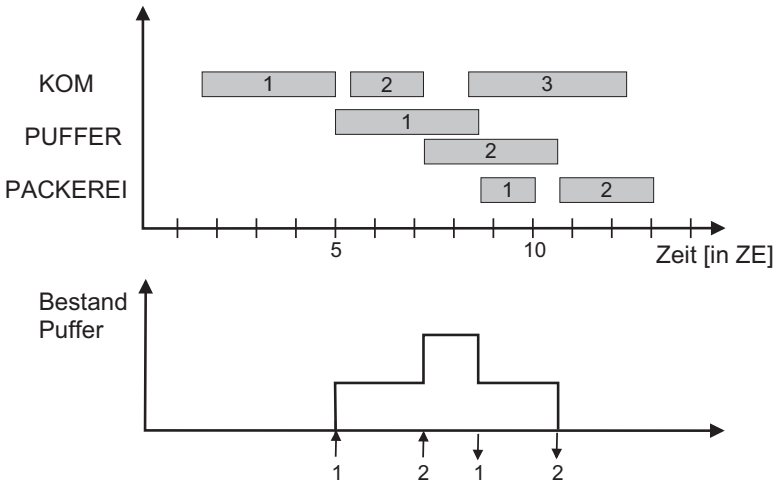


Abbildung 4.5.: Entwicklung des Pufferbestandes, abhängig von den vor- und nachgelagerten Operationen der Aufträge 1,2 und 3.

Diese Gleichung läßt sich zusammenfassen zu

$$\sum_{O_i^j | (O_k^j \succ O_i^j \succ O_g^j) \wedge (t_e(O_k^j) < t < t_s(O_g^j))} sz(O_i^j) \leq cp(p) \quad \forall\, ra(O_i^j) = p \tag{4.12}$$

Alle Produkte, deren Operationen auf der Vorgänger-Ressource vor t beendet wurden und deren Bearbeitung nach t auf der Nachfolger-Ressource begonnen wird, befinden sich im Puffer. Bedingung (4.12) gilt, wenn der Puffer zu Beginn und am Ende des Untersuchungszeitraumes leer ist. Da $t^{rel}(j)$ und $t^{due}(j)$ jedes Auftrages $j \in \mathcal{J}$ in dem Untersuchungszeitraum $0, \ldots, H$ liegen, ist diese Bedingung erfüllt.
Eine zusätzliche Übergangszeit, beispielsweise um Produkte zwischen der einlagernden, bzw. auslagernden Ressource zum Puffer zu transportieren, könnte über eine zusätzliche Ressource integriert werden. Das erzeugt einen unnötigen zusätzlichen Aufwand, auf Seite 34 wird eine allgemeine Formulierung von Übergangszeiten angegeben.

Ressourcen-Pools

Bei einem mehrstufigen (nicht automatischen) Kommissioniersystem werden die Kommissionier-Ressourcen[6] von Personal bedient. Steht kein Personal zur

[6] Bei statischer Bereitstellung können das Hilfsmittel wie Kommissionier-Fahrzeuge, bei dynamischer Bereitstellung der physische Kommissionierplatz sein.

Verfügung, kann nicht kommissioniert werden. Steht keine Kommissionier-Ressource zur Verfügung, ist ebenfalls kein Kommissionieren möglich. Es besteht die Möglichkeit, Personalpools zu bilden, die unterschiedliche Bereiche (Wareneingang, Kommissionierung, Packerei) versorgen[7]. Personal stellt also auch eine (bearbeitende) Ressource mit beschränkter Kapazität dar. Um den Kommissioniervorgang auszuführen, sind somit zwei Ressourcen nötig.

Zur Ausführung einer Operation ist eine Untermenge der vorhandenen Ressourcen nötig, die zu einer *Menge* von Ressourcen zusammengefaßt werden $rs \in \mathcal{RS}$, wobei $\mathcal{RS} \subseteq \mathcal{R}$. Alle benötigten Ressourcen müssen über den Ausführungszeitraum der Operation zur Verfügung stehen, die gegebenen Kapazitäten dürfen nicht überschritten werden, Constraint (4.10) muß erfüllt sein.

Von Nuijten (1994) wird eine Modellierung von Ressourcen-Mengen vorgeschlagen, die für die Modellierung von Personal oder anderer Pool-Ressourcen zu restriktiv ist. Hier dürfen sich zwei Operationen, die auf zwei Mengen von Ressourcen eingeplant werden und die gleiche Elemente enthalten, nicht überlappen:

$$\begin{aligned} & (ra(O_i^j) \cap ra(O_k^l) \neq \emptyset \wedge ra(O_i^j) \neq ra(O_k^l)) \\ \Rightarrow \quad & (t_e(O_i^j) \leq t_s(O_k^l) \vee t_e(O_k^l) \leq t_s(O_i^j)) \end{aligned} \tag{4.13}$$

In der Praxis können gültige Lösungen erzeugt werden, die Gleichung (4.13) nicht erfüllen, etwa bei dem angesprochenen Fall bei (Personal-)Ressourcen mit $cp(r) > 1$. Ein Teil des Personals kann im Kommissionierbereich, ein weiterer Teil in der Packerei und ein Mitarbeiter im Wareneingang eingesetzt werden.

Bei den hier untersuchten mehrstufigen Umschlagsystemen ist die Verwendung von Ressourcen-Pools sogar *nur* dann sinnvoll, wenn es Überschneidungen der Mengen gibt. Ein Mitarbeiter, der immer einer bestimmten Ressource zugeordnet ist, muß nicht als Ressourcen-Pool modelliert werden. Die Ressource ist nur verfügbar, wenn der Mitarbeiter verfügbar ist, somit lassen sie sich zusammenfassen.

Die angegebenen Kapazitätsconstraints dürfen zu keinem Zeitpunkt verletzt werden. Sei der Pool mit r_p, und die übrigen Ressourcen mit $r_1, \ldots, r_n$ bezeichnet. Die Mengen von Ressourcen ergeben sich zu $rs_i = r_i \cup r_p,\ i = 1, \ldots, n$. Also ist die Pool-Ressource in jeder Menge enthalten. Die Definition des Ressourcen-Bedarfes wird erweitert zu $sz : \mathcal{O} \times \mathcal{R} \rightarrow \mathbb{N}^+$. Das Kapazitätsconstraint (4.10) muß für die

[7] Hierbei wird vorausgesetzt, daß das Personal auch in diesen unterschiedlichen Bereichen einsetzbar ist.

beiden Ressourcen r_i und r_p zu jedem Zeitpunkt erfüllt sein:

$$\sum_{o|ra(o)=rs_i, r_i \in rs_i \wedge t_s(o) \leq t \leq t_e(o)} sz(o, r_p) \leq cp(r_i, t)$$
$$\sum_{o|ra(o)=rs_i, r_p \in rs_i \wedge t_s(o) \leq t \leq t_e(o)} sz(o, r_p) \leq cp(r_p, t) \tag{4.14}$$

Gilt $cp(r_i) = 1, \ \forall \ i = 1, \ldots, n$ wird entsprechend Constraint (4.9) verwendet, auch mit der Erweiterung um Übergangszeiten, die im folgenden eingeführt werden.

Alternative Ressourcen

Stehen zur Ausführung einer Operation mehrere Ressourcen zur Verfügung, kann eine *alternative* Zuweisung erfolgen. Nach Dörrsam (1999) kann unterschieden werden in

- funktionshomogene, bzw. funktionsinhomogene,
- zeithomogene, bzw. zeitinhomogene

alternative Ressourcen.
Die Zuweisung für den kurzfristigen Fall ist recht komplex. Kapazitäts-Constraints lassen sich nur als obere Schranke angeben, da die reihenfolgeabhängigen Übergangszeiten nicht a-priori bestimmt werden können. Es ergibt sich das Dilemma der simultanen Zuweisung und Reihenfolgebildung. In Kapitel 7 werden alternative Ressourcen zur Modellierung und zum Lösen eines komplexen Reihenfolgeproblems verwendet.

Übergangszeiten

Werden bei einem Umschlagsterminal zwei Container mit demselben Kran k_1, Kapazität $cp(k_1) = 1$, von einem Zug auf zwei Lkw umgeschlagen, kann das mit Hilfe von zwei Operationen O_1^1, O_1^2 modelliert werden, für die das Kapazitäts-Constraint (4.9) erfüllt sein muß. Nach dem Absetzen des ersten Containers ist eine Leerfahrt nötig, um zur Aufnahme-Position des zweiten Containers zu fahren. Die Leerfahrzeit t^{leer} ist je nach realisierter Reihenfolge ($O_1^1 \succ O_1^2$, bzw. $O_1^2 \succ O_1^1$) unterschiedlich, also *reihenfolgeabhängig*. In der Scheduling-Literatur werden diese Leerfahrten als reihenfolgeabhängige Rüstzeiten, *sequence-dependent setup time* bezeichnet. Übergänge können auftreten durch:

- Leerfahrten von einem Absetzpunkt zu einem Aufnahmepunkt,
- Umrüstvorgänge (Werkzeugwechsel, Handling, etc.),
- Eine minimale Zeitlücke beim Einschleusen auf Förderstrecken,
- Allgemeine Beziehungen zwischen zwei Operationen, beispielsweise der Transport von einer bearbeitenden Ressource zu einer puffernden.

Mit $t^{set}(O_i^j, O_k^l)$ wird wiedergegeben, wieviel Zeit zwischen dem Ende von Operation O_i^j und dem Start von O_k^l verstreichen muß, bevor die Ressource $(ra(O_i^j) = ra(O_k^l))$ die Operation O_k^l ausführen kann. Die Reihenfolgeabhängigkeit, also $t^{set}(O_i^j, O_k^l) \neq t^{set}(O_k^l, O_i^j)$ ist in Abbildung 4.6 anhand eines Umschlagproblems dargestellt. Im Produktionsbereich kann z.B. ein Farbwechsel in der Lackiererei von Weiß nach Schwarz schneller sein als umgekehrt, da keine Reinigung der Düsen nötig ist. Es muß gelten:

$$t_e(O_i^j) + t^{set}(O_i^j, O_k^l) \leq t_s(O_k^l) \tag{4.15}$$

Würde das $\leq$ durch ein $=$ ersetzt, wären die beiden Operationen fest aneinander gebunden. Durch das $\leq$ können auch mininimale Übergangszeiten modelliert werden, was für die im folgenden dargestellte Kombination von puffernden und bearbeitenden Ressourcen wichtig ist.

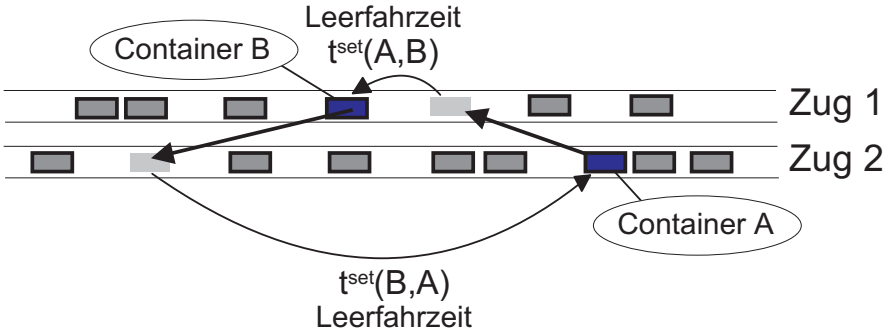


Abbildung 4.6.: Verdeutlichung der reihenfolgeabhängigen Übergänge. Zwei Container sollen zwischen zwei Zügen umgeschlagen werden, abhängig von der Reihenfolge der Operationen A $\rightarrow$ B oder B $\rightarrow$ A, ergeben sich deutlich unterschiedliche Übergänge (d.h. Leerfahrten).

Neben den Übergangs*zeiten* lassen sich auch Übergangs*kosten* angeben. In der Literatur finden sich zahlreiche Arbeiten, die die Kosten in die Zielfunktion aufnehmen, jedoch die Zeiten nicht in die Nebenbedingungen integrieren. Diese Ansätze

sind für praxisrelevante Fragestellungen nicht anwendbar, da durch die Übergangszeiten Kapazitäten gebunden werden. Durch Nichtbeachtung dieser Tatsache werden die in den Gleichungen (4.9) und (4.10) dargestellten Bedingungen verletzt, es entstehen per se ungültige Lösungen (siehe auch Kapitel 3.5 und Abbildung 4.7).

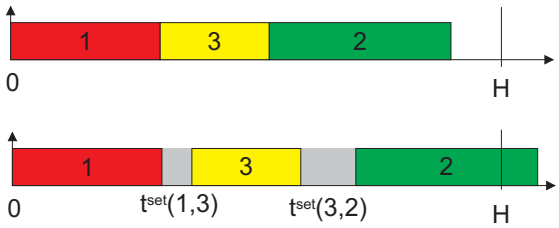


Abbildung 4.7.: Bei Vernachlässigung der Übergangszeiten (oben) wäre die hier gezeigte Reihenfolge von Operationen in dem Zeitraum $1, \ldots, H$ gültig. Die Beachtung der Rüstzeiten (unten) zeigt jedoch eine Verletzung des erlaubten Zeitraumes.

Die Berücksichtigung von reihenfolgeabhängigen Übergangszeiten ist nur sinnvoll, wenn zwei Operationen um die gleiche Ressource $r \in \mathcal{R}$ mit $cp(r) = 1$ konkurrieren. Um die Zeiten zu beachten, wird die Kapazitätsbedingung (4.9) erweitert. Es ergibt sich:

$$\begin{aligned} &t_e(O_i^j) + t^{set}(O_i^j, O_k^l) \leq t_s(O_k^l) \ \vee \\ &t_e(O_k^l) + t^{set}(O_k^l, O_i^j) \leq t_s(O_i^j) \quad \forall\, ra(O_k^l) = ra(O_i^j) \end{aligned} \tag{4.16}$$

Identische Ressourcen können zu einer Ressource mit $cp(rs) > 1$ zusammengefaßt werden[8]. Für den Fall, daß keine (oder vernachlässigbar kleine) reihenfolgeabhängigen Übergangszeiten zu berücksichtigen sind, ist diese Modellierung zulässig. Andernfalls muß die Gruppe in die einzelnen Ressourcen aufgeteilt werden, um eine gültige Lösung zu erhalten, hier wird wiederum (4.16) verwendet.

Sind zwei zu bearbeitende Ressourcen über einen Puffer entkoppelt, und ist der Transport zwischen den Ressourcen und dem Puffer kapazitativ unkritisch, kann die Transportzeit als Übergangszeit modelliert werden. Die Kapazität des Puffers wird nach dem Hintransport und vor dem Weitertransport belastet. Die Aufenthaltszeit im Puffer hängt wiederum von den Vorgänger- und Nachfolger-Operationen ab. Zwei Operationen O_k^j und O_g^j sind durch eine Operation O_j^i auf der puffernden Ressource entkoppelt. Die Transportzeit zwischen der ersten Operation und dem Puffer ist $t^{set}(O_k^j, O_i^j)$, der Transport zwischen dem Puffer und

[8] Eine Ausnahme bilden Ressourcen, die nur produktspezifisch belegt werden dürfen, wie beispielsweise Tanks oder Silos in der chemischen Industrie. Diese Ressourcen müssen getrennt voneinander betrachtet werden, es gilt weiterhin $cp(r) = 1$.

der zweiten Operation benötigt $t^{set}(O_i^j, O_g^j)$. Gleichung (4.12) läßt sich somit erweitern zu:

$$\begin{aligned} & t_e(O_k^j) + t^{set}(O_k^j, O_i^j) + t^{set}(O_i^j, O_g^j) \leq t_s(O_g^j) \\ & \sum_{O_i^j \mid \alpha} sz(O_i^j) \leq cp(p) \forall\ ra(O_i^j) = p, \quad 0 \leq t \leq H \\ & \alpha \leftrightarrow (O_k^j \succ O_i^j \succ O_g^j) \wedge (t_e(O_k^j) + t^{set}(O_k^j, O_i^j) < t < t_s(O_g^j) - t^{set}(O_i^j, O_g^j)) \end{aligned} \tag{4.17}$$

Kombination bearbeitender- und puffernder Ressourcen

Die Kombination einer bearbeitenden und puffernden Ressource liegt beispielsweise bei einem fahrerlosen Transport-Fahrzeug (FTF) vor. Soll eine Fördereinheit von Ressource A zu B transportiert werden, ist das FTF eine bearbeitende Ressource, und die Fahrzeit entspricht der Bearbeitungszeit. Kann die Fördereinheit nicht sofort an Ressource B übergeben werden, ist das FTF noch belegt und wird zur puffernden Ressource, bis die Operation weiterbearbeitet wird (siehe Abbildung 4.8). Zwei Operationen eines Auftrages $O_k^j \succ O_g^j$ sind über ein FTF verbunden, O_i^j ist die Kombinations-Operation aus Transport und Pufferung. Stellt $pt(O_i^j)$ die Fahrzeit dar, so kann frühestens zum Zeitpunkt $t_e(O_k^j) + pt(O_i^j)$ mit der Bearbeitung von O_g^j begonnen werden. Die puffernde Ressource FTF ist jedoch bis zu dem realen Start von O_g^j, also $t_s(O_g^j)$ belegt. Somit muß Gleichung (4.12) erfüllt sein (siehe auch Abbildung 4.8):

$$\begin{aligned} & t_e(O_k^j) + pt(O_i^j) \leq t_s(O_g^j) \\ & \sum_{O_i^j \mid (O_k^j \succ O_i^j \succ O_g^j) \wedge (t_e(O_k^j) < t < t_s(O_g^j))} sz(O_i^j) \leq cp(p) \quad \forall\ ra(O_i^j) = p,\ 0 \leq t \leq H \end{aligned} \tag{4.18}$$

In Kapitel 6 und 7 wird diese Modellierung verwendet.

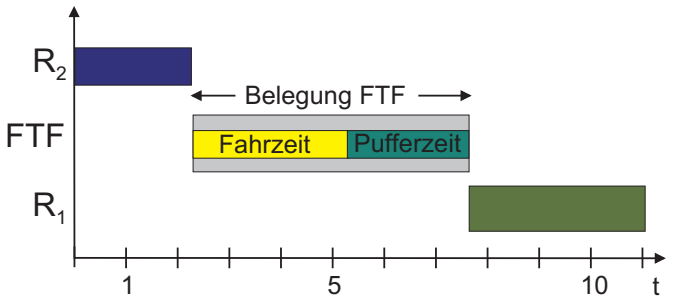


Abbildung 4.8.: Kombination von puffernder und transportierender Ressource (FTF). R_1 und R_2 sind beliebige Ressourcen. Dargestellt sind die Fahrzeit des FTF, die Pufferzeit und die gesamte Belegungsdauer.

4.3. Mehrstufige Systeme

Auf einer Stufe können n parallele Maschinen arbeiten. Sind zur Bearbeitung der Aufträge mehrere Stufen nötig, z.B. die Auslagerung, Kommissionierung, Packerei und der Versand in einem Distributionszentrum, spricht man von einem *mehrstufigen System*. Der Hauptunterschied zu einstufigen Systemen ist die Abhängigkeit zwischen den Operationen der Stufen. Sind Fertigstellungszeitpunkte gegeben, wirken sich diese auf alle durch Reihenfolge-Constraints gebundenen Operationen auf die vorangehenden Stufen aus.

Der Unterschied zum Spezialfall des flow-shop Problems liegt darin, daß Flüsse durch das System, d.h. die Reihenfolge der Ressourcen, beliebig sein können. Ressourcen sind durch Puffer entkoppelt, die Flußrichtung der Aufträge und damit die Bearbeitungsreihenfolge der Operationen ist nicht vorgegeben. Weiterhin sind reihenfolgeabhängige Übergangszeiten, Mengen von Ressourcen und alternative Ressourcen zu berücksichtigen.

4.4. Zusammenfassung

Die Lösung eines Optimierungsproblems gliedert sich grob in die Schritte Modellierung - Formulierung - Implementierung. Die Modellierung legt den Aggregationsgrad und damit zum großen Teil die Laufzeit des formulierten und implementierten Modells fest. Durch eine geschickte Modellierung kann die Laufzeit zur Bestimmung einer (sub-) optimalen Lösung deutlich gesteigert werden.

Die in diesem Kapitel vorgestellte Modellierung basiert auf der Systemtheorie. Die Elemente eines mehrstufigen Umschlagsystems, Operationen und Ressour-

cen sowie Aufträge werden eingeführt und definiert. Die Beziehungen werden in temporale, Reihenfolge- und kapazitative Constraints unterteilt. Es werden Bedingungen angegeben, die bei einer gültigen Lösung erfüllt sein müssen. Es werden puffernde und bearbeitende Ressourcen eingeführt, die temporalen Beziehungen werden um reihenfolgeabhängige Übergangszeiten erweitert. Anschließend werden mehrstufige Systeme vorgestellt und kurz auf die Wahl des richtigen Aggregationsgrades eingegangen.

In diesem Kapitel wurde somit der Grundstein gelegt, um ein mehrstufiges Umschlagsystem zu modellieren und als Constraint Optimization Problem zu formulieren.

5. Constraint Satisfaction

„Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.“ (Eugene C. Freuder, CONSTRAINTS, April 1997)

Das Konzept der Constraints hat eine sehr lange Tradition in der Mathematik. Eines der ältesten Probleme besteht im Lösen von linearen Gleichungssystemen, prinzipiell ein sog. *Constraint Satisfaction Problem*. In jüngster Zeit wurden Programmiersprachen im Bereich des Constraint Programming mit kommerziellem Erfolg konsequent weiterentwicklt. Der Verband ACM (Association for Computing Machinery) sieht Constraint Programming als eine der strategischen Ausrichtungen der zukünftigen Forschung. Auch Pountain (1995) bezeichnet Constraint Programming (CP) als das Paradigma, das künftig den größten kommerziellen Erfolg haben wird. Schwere kombinatorische Probleme können (relativ) einfach formuliert und mit leistungsfähigen Algorithmen gelöst werden. Mackworth (1987) spricht davon, daß Wissen am besten dargestellt werden kann als „was ist erlaubt“ und „was ist verboten“. Er kritisiert, daß ein zu lösendes Problem bei herkömmlichen Systemen/Verfahren zu sehr in deren *Sprache* übersetzt werden muß. Die entstehenden Modelle sind „überspezifiziert“, was zu einer unnötigen Komplexität und Redundanz führt.

Ein Constraint Satisfaction Problem (CSP) besteht darin, jeder gegebenen Variablen einen Wert zuzuweisen, so daß eine Menge von Constraints erfüllt ist. Durch Erweiterung entstehen Constraint Optimization Probleme (COP), hier reicht die Gültigkeit einer Lösung nicht aus, es soll die komplexere Frage nach der Optimalität gelöst werden.

In Kapitel 5.1 wird das CSP formal definiert. Das CSP läßt sich lösen, indem die Variablen in einer gegebenen Reihenfolge ausgewählt und mit einem bestimmten Wert instantiiert werden. Werden hierdurch Constraints verletzt, muß eine oder mehrere der letzten Zuweisungen rückgängig gemacht werden. Dieser Vorgang wird als *Backtracking* bezeichnet und sollte im Sinne eines effizienten Suchprozesses vermieden werden, Ansätze dazu werden in Kapitel 5.2 beschrieben. In Kapitel

5.3 werden Konzepte zur Einschränkung des Suchraumes vorgestellt, insbesondere Verfahren der Konsistenzprüfung und -sicherstellung. Unterschiedliche Verfahren zur Auswahl von Variablen und Werten und zur Verbesserung des Suchprozesses durch Vermeidung von backtrack-Schritten werden in Kapitel 5.4 erläutert. Die Erweiterung eines CSP in ein COP wird in Kapitel 5.5 behandelt, hier werden die in den Anwendungskapitel genutzten Suchverfahren eingeführt.

5.1. Constraint Satisfaction Problem

Durch den generischen Ansatz können viele unterschiedliche Fragestellungen als Constraint Satisfaction Probleme modelliert und gelöst werden. Einen Überblick gibt u.a. Prosser (1993). Ein Constraint Satisfaction Problem wird formal wie folgt definiert:

Definition 3 *Ein* Constraint Satisfaction Problem (CSP) *ist gegeben als ein Tripel* (X, D, C)*, wobei* $X = \{x_1, \ldots, x_n\}$ *eine Menge von Variablen darstellt,* $D = \{D_1, \ldots, D_n\}$ *eine Menge von Wertebereichen, so daß für jede Variable* x_i *ein Wertebereich* D_i *gegeben ist und* $C = \{c_1, \ldots, c_m\}$ *eine Menge von Constraints. Jedes Constraint ist auf einem Tupel von Variablen definiert. Die Frage ist, ob es eine Zuweisung* $a \in D_1 \times \ldots \times D_n$ *gibt, so daß alle Constraints* $c_i(a)$*,* $1 \leq i \leq m$ *erfüllt sind. Eine solche Zuweisung wird auch* Lösung *des CSP genannt.*

Die Variablen sind die zu ermittelnden Größen des Problems, sie können beliebige numerische oder alphanumerische Werte annehmen und sind durch Wertebereiche determiniert. Die Wertebereiche können reale oder ganze Zahlen, Buchstaben, Wörter, oder Mengen sein, z.B. $D_a = \{1, 2, 4, 5, 8, 120\}$, $D_b = \{A, B, C, D\}$, $D_c = \{1, \ldots, 10\}$ oder $D_d = \mathbb{R}^+$. Ist die Größe aller Wertebereiche $|D|$ abzählbar endlich, spricht man von *finiten* Wertebereichen. Constraints können einwertig $c_1 : x \leq 5$, zweiwertig (binär) $c_2 : x + y \leq 7$ oder n-wertig $c_3 : \sum_{i=1}^{n} x_i \leq 5$ sein. Im Bereich der kombinatorischen Optimierung kommen hauptsächlich ein- und zweiwertige Constraints zum Einsatz. Wird ein Constraint oder eine Kombination von Constraints ausgewertet, erhält man die (boolsche) Aussage *wahr* oder *falsch*. Eine kontinuierliche Bewertung auf der Basis von Fuzzy-Logic wird von Schiex (1997) eingeführt.

BEISPIEL: Die Variablen seien x, y, z, die Wertebereiche $D_x = \{0, \ldots, 10\}$, $D_y = \{3, \ldots, 7\}$, $D_z = \{1, \ldots, 4\}$ und als Constraints $c_1 : x + y < 5$, $c_2 : x < z$ sowie die Forderung, daß alle Werte unterschiedlich sind (was sich als $c_3 : x \neq y$, c_4 :

$x \neq z,\ c_5 : y \neq z$ beschreiben läßt). Eine mögliche Zuweisung, die die gegebenen Constraints erfüllt, ist durch $a = (0, 3, 1)$ gegeben.

Werden die Variablen als die Knoten und die Constraints als Kanten eines Graphen $G = (X, C)$ dargestellt, erhält man einen sogenannten *Constraint Graph* (siehe Abbildung 5.1).

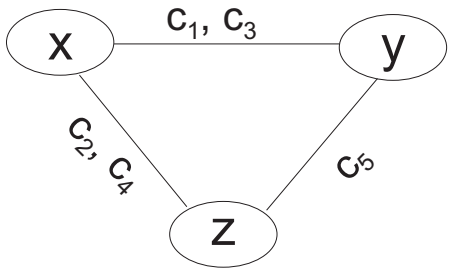


Abbildung 5.1.: Constraint-Graph, bestehend aus den Variablen (Knoten) und den Constraints (Kanten), die auf den Variablen definiert sind.

Constraints können beliebig komplex sein und auch weitere Constraints enthalten. Die Forderung, daß alle Werte unterschiedlich sein sollen, kann beispielsweise auch als $c_6 : c_3 \vee c_4 \vee c_5 \Leftrightarrow (x \neq y) \vee (x \neq z) \vee (y \neq z)$ geschrieben werden. Bei sogenannten *Meta-Constraints* werden, abhängig von der Erfüllung, bzw. Nichterfüllung eines Constraint (c_a) weitere Constraints (c_b bzw. c_c) hinzugefügt. Soll bei Erfüllung von Constraint c_a auch c_b erfüllt sein, so gilt: $c_m : c_a \wedge c_b$. Soll bei Nichterfüllung von c_a das Constraint c_c erfüllt werden, ergibt sich $c_n : \neg c_a \wedge c_c$. Kombiniert erhält man:

$$c_m : (c_a \wedge c_b) \vee (\neg c_a \wedge c_c) \tag{5.1}$$

In der Literatur, unter anderem bei Fox (1994), wird unterschieden zwischen *nicht relaxierbaren* (harten) Constraints, die bei einer gültigen Lösung erfüllt sein müssen und *relaxierbaren* (weichen) Constraints, die erfüllt sein sollten. Beispiele für nicht relaxierbare Constraints sind Reihenfolge-Beziehungen im Vorranggraphen. So muß ein Artikel in einem Distributionszentrum kommissioniert werden, bevor er verpackt werden kann. Ein relaxierbares Constraint ist beispielsweise die Einhaltung eines vorgegebenen Wunschtermins. Wird dieser überschritten, kann der Auftrag noch (verspätet) ausgeliefert werden, d.h. der Wunschtermin muß relaxiert (angepaßt) werden, damit sich das Constraint erfüllen läßt.

Das CSP ist, wie von Nuijten (1994) gezeigt wurde, NP-vollständig. Die Lösung des CSP ist Gegenstand zahlreicher Forschungsarbeiten, Nadel (1989) und Kumar (1992) geben einen guten Überblick. Jedes CSP kann nach Shapiro (1987)

formuliert werden als „Finde eine Menge von Werten der Variablen, die die gegebenen Constraints erfüllen“. D.h. finde den optimalen Punkt in dem Suchraum $D_1 \times D_2 \times \ldots \times D_n$, bzw. eine Menge von Punkten bei mehreren gültigen Lösungen. Ein einfacher Ansatz ist es, alle (theoretisch möglichen) Kombinationen von Wertezuweisungen zu untersuchen. Die Größe des Suchraumes wäre somit $|D_1| \times |D_2| \times \ldots \times |D_n|$. Offensichtlich ist dieses Vorgehen sehr zeit-, bzw. rechenintensiv.

5.2. Chronological Backtracking und Erweiterungen

Ein sehr ineffizientes Verfahren zur Lösung eines CSP ist das sog. *generate-and-test*. Hier werden *zuerst* allen Variablen Werte zugeweisen. *Anschließend* wird die Gültigkeit der Lösung überprüft. Dieses Verfahren durchsucht einen weitaus größeren Bereich des Suchraumes als die im folgenden dargestellten. Einen guten Einblick in die Suchverfahren geben Shanahan und Southwick (1989).

In Algorithmus 1 ist der einfachste Lösungsalgorithmus für ein CSP, das sogenannte *chronological backtracking* dargestellt. Die Idee ist, den Lösungs-Vektor $a = (x_1, \ldots, x_n)$ elementweise aufzubauen und bei Hinzufügen eines Elementes zu prüfen, ob eine partielle Lösung $a_p = (x_1, \ldots, x_k)$ noch zu einer gültigen Lösung führen kann. Wird ein Konflikt aufgedeckt, d.h. ist der Wertebereich D_i einer Variablen $x_i \in a_p$ leer, $D_i = \emptyset$, so muß die letzte (oder die letzten n) Zuweisung(en) rückgängig gemacht werden. Von diesem Schritt hängt entscheidend die Effizienz des Verfahrens ab. Das Verfahren ist exakt, d.h. es findet die optimale Lösung (der Suchraum wird unter Umständen vollständig enummeriert). Häufig soll in kürzester Zeit eine gültige, möglichst gute Lösung erzeugt werden. Dieses Ziel wird mit den in den folgenden Kapiteln beschriebenen Ansätzen verfolgt.

Eine Eigenschaft des Algorithmus, die die Suche sehr ineffizient werden läßt, wird als *trashing* bezeichnet. Sehr große Teile des Suchraums werden überflüssigerweise untersucht, da ein Konflikt eines späteren Schrittes immer wieder auftritt. Der Konflikt könnte nur durch Änderung eines sehr frühen Suchschrittes vermieden werden.

BEISPIEL: Die Startzeitpunkte der Operationen x_1 und x_2 haben den gleichen Wertebereich $D_1 = D_2 = \{0, 1, 2\}$, die Variable x_k kann Werte aus $D_k = \{0, 1\}$ annehmen. Die restlichen Variablen $x_3, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n$ haben den Wertebereich $D = \{0, \ldots, 1000\}$. Die Dauer der Operationen sei $pt(1) = pt(2) = pt(k) = 1$, die verfügbare Ressource habe eine Kapazität von eins. Werden die Variablen

Algorithmus 1 Chronological Backtracking mit Erweiterungen

```
1:  while nicht gelöst do
2:    Konsistenz-Prüfung (und Sicherstellung)
3:    if Konflikt then
4:      (look-back Verfahren)
5:      backtrack-Schritt
6:    else
7:      (Anwendung look-ahead Verfahren)
8:      Auswahl Variable
9:      Auswahl Wert
10:   end if
11: end while
```

der Reihe nach ausgewählt und immer der kleinstmögliche Wert zugewiesen, wird zuerst $x_1 = 0$ und $x_2 = 1$ gesetzt. Nun wird mit den Variablen $x_3, \ldots, x_{k-1}$ weiter verfahren, bis bei der Zuweisung von x_k ein Konflikt auftritt, da beide Startzeitpunkte bereits belegt sind. Die letzte Zuweisung x_{k-1} wird rückgängig gemacht, ein neuer Wert zugewiesen und wieder der Konflikt bei x_k aufgedeckt. Sind alle Werte von x_{k-1} überprüft, wird die Zuweisung von x_{k-2} rückgängig gemacht, usw. Der Teil des Suchraums mit den Variablen $x_3, \ldots, x_{k-1}$ wird ohne Ergebnis vollständig durchsucht, bis die *richtigen* Variablen erreicht sind. Wenn nicht die Zuweisung der zuletzt instantiierten Variable, sondern der Variable, die den Konflikt ausgelöst hat, zurückgesetzt wird, kann der Konflikt sehr viel früher gelöst werden. Beispielsweise liefert $x_2 = 2, \ldots, x_k = 1$ eine gültige Zuweisung. D.h. es ist eine Reihenfolge zu bestimmen, so daß bei der Instantiierung eine minimale Anzahl von Konflikten erzeugt wird. Auf diese *Sortierung* der Variablen und Werte wird in Kapitel 5.4 eingegangen.

Weiterentwicklungen wie *backjumping*, *intelligent backtracking*, Anwendung von *Lern-Verfahren* und *Schnitt-Verfahren* werden in Dechter (1990) behandelt. Das Ziel aller Ansätze ist es, möglichst viel über die Struktur des Suchraumes herauszufinden, um eine Suche ohne backtrack-Schritt zu ermöglichen. Durch Konsistenz-Prüfung und Sicherstellung werden die Werte der Variablen entfernt, die nicht in einer Lösung vorkommen können. Vorausschauende (look-ahead) Verfahren identifizieren Variablen, die backtrack-Schritte überflüssig machen, wohingegen look-back Verfahren das Ziel haben, einen aufgetretenen Konflikt möglichst effizient zu lösen.

5.3. Einschränkung des Suchraumes

Nach der Einschränkung des Suchraumes kann wesentlich effizienter nach Lösungen gesucht werden. Nur durch Anwendung von Algorithmen zur Einschränkung des Suchraumes konnten Caseau und Laburthe (1995) für 23 von 40 Benchmark-Problemen die Optimalität beweisen, es war keine Suche mehr nötig. Hier ist immer zwischen dem Aufwand zur Einschränkung des Suchraumes und dem Aufwand zur Suche auf dem weniger eingeschränkten Suchraum abzuwägen. Die Algorithmen sind sinnvoll zur Bestimmung von unteren und oberen Schranken.

In den folgenden Kapiteln werden unterschiedliche Ansätze zur Einschränkung des Suchraumes beschrieben und zur Anwendung auf die untersuchten Umschlagsprobleme erweitert bzw. angepaßt.

5.3.1. Konsistenz-Prüfung und Sicherstellung

Eine Menge von Variablen wird als konsistent bezeichnet, wenn kein Constraint durch eine beliebige Wertekombination verletzt wird. Eine logische Konsequenz nach der Prüfung (checking) der Konsistenz ist die Nutzung des Wissens zur Sicherstellung (enforcing) der Konsistenz durch Einschränken der Wertebereiche. Hierbei wird unterschieden zwischen Knoten-, Kanten-, und Pfad (k-)Konsistenz.

Definition 4 *Ein Wertebereich D_x einer Variablen x ist für ein gegebenes Constraint c_k* knoten-konsistent*, wenn $c_k(v)$ für alle $v \in D_x$ erfüllt ist.*

BEISPIEL: $D_x = \{1, \ldots, 5\}$, $c_k : x < 6$ ist knoten-konsistent, da das Constraint c_k für jeden Wert x aus D_x erfüllt ist.

Die Knoten-Konsistenz kann sehr leicht sichergestellt werden, indem alle Werte, die das Constraint nicht erfüllen, aus dem Wertebereich entfernt werden.

Wird die Definition auf zweiwertige Constraints übertragen, läßt sich die *Kanten*-Konsistenz definieren.

Definition 5 *Seien x und y Variablen mit den Wertebereich D_x und D_y und c_l ein Constraint, das auf den beiden Variablen definiert ist. Der Wertebereich D_x ist* kanten-konsistent *zu D_y, wenn es für jeden Wert $v \in D_x$ einen Wert $w \in D_y$ gibt, so daß c_l gültig ist.*

Die Definition ist direktional, d.h. ist D_x mit D_y kanten-konsistent, muß nicht D_y mit D_x kanten-konsistent sein. Die Konsistenz kann erreicht werden, indem alle Werte $v \in D_x$, für die kein Wert aus D_y gefunden werden kann, so daß c_l

gültig ist, aus dem Wertebereich entfernt werden. Dieser Schritt ist zulässig, da die Werte in keiner gültigen Lösung enthalten sein können.

Die Erweiterung auf n-wertige Constraints wird als Pfad- oder k-Konsistenz bezeichnet. Kumar (1992) gibt eine verbale Definition: Wähle Werte von $k-1$ beliebigen Variablen, die alle Constraints zwischen diesen Variablen erfüllen. Nun wähle die k-te Variable. Falls ein Wert existiert, der alle Constraints zwischen den k Variablen erfüllt, ist k-Konsistenz erreicht.

Die Sicherstellung der k-Konsistenz ist sehr rechenaufwendig, jedoch soll an dem folgenden Beispiel gezeigt werden, daß dieser Aufwand sinnvoll sein kann.

BEISPIEL: Gegeben sind die Operationen x, y, z mit den Wertebereichen der Startzeitpunkte $D_x = D_y = D_z = \{0, \ldots, 10\}$, den Bearbeitungszeiten $pt(x) = pt(y) = pt(z) = 2$, eine vorgegebene Reihenfolge $x \succ y \succ z$ und damit die Constraints $c_1 : t_s(x) + pt(x) \leq t_s(y)$, $c_2 : t_s(y) + pt(y) \leq t_s(z)$. Wird nun Kanten-Konsistenz in der Reihenfolge D_x, D_y, D_z sichergestellt, so ergeben sich die folgenden (angepaßten) Wertebereiche:

$$\begin{aligned} \text{Konsistenz für } (D_x, D_y) : D_x &= \{0, \ldots, 8\},\ D_y = \{2, \ldots, 10\} \\ \text{Konsistenz für } (D_y, D_z) : D_y &= \{2, \ldots, 8\},\ D_z = \{4, \ldots, 10\} \end{aligned}$$

Durch Anwendung der Pfad-Konsistenz werden die Wertebereiche weiter eingeschränkt zu

$$D_x = \{0, \ldots, 6\},\ D_y = \{2, \ldots, 8\},\ D_z = \{4, \ldots, 10\}$$

$KON(D_1, \ldots, D_n)$ bezeichnet im folgenden den Algorithmus zur Sicherstellung der Konsistenz. Bei $n = 1$ handelt es sich um Knoten-, bei $n = 2$ um Kanten-, und bei $n > 2$ um Pfad-Konsistenz.

Die Überprüfung und Sicherstellung der Konsistenz sollte für alle Variablen durchgeführt werden, auf denen Constraints definiert sind. Die Reihenfolge, in der die Variablen untersucht werden, ist für die Effizienz der Konsistenzsicherstellung wichtig. Sind für mehrere Variablen Reihenfolge-Beziehungen definiert, bilden die Variablen also eine *Kette*, so sollten die Variablen in der Reihenfolge der Kette ausgewählt werden. Auf den Variablen v, w, x, y, z sind die Reihenfolge-Beziehungen $c_1 : v \succ w$, $c_2 : w \succ x$, $c_3 : x \succ y$ und $c_4 : y \succ z$, zusammengefaßt $v \succ w \succ x \succ y \succ z$ definiert. Die zugrunde liegende Kette, die die Reihenfolge der Auswahl zur Sicherstellung der Konsistenz definiert, ist mit $k = \{v, w, x, y, z\}$ gegeben.

Eine ungünstige Reihenfolge wäre $k_u = \{z, y, x, w, v\}$. Hier ist nach Anwendung von $KON(D_z, D_y)$ und $KON(D_y, D_x)$ erneut $KON(D_z, D_y)$ nötig, da sich durch

$KON(D_y, D_x)$ der Wertebereich D_y evtl. geändert hat, was wiederum Auswirkungen auf D_z hat.

Idealerweise läßt sich eine Kette bilden, die alle zu instantiierenden Variablen enthält. Die *implizit* vorhandenen Reihenfolgebeziehungen werden somit *explizit*. Kann eine solche Sortierung erreicht werden, ist die Lösung des Problems trivial. Bei praktischen Anwendungsfällen ist eine solche globale Kette jedoch eine Ausnahme.

Konsistenz-Prüfung und Sicherstellung für mehrstufige Umschlagsysteme

Im den folgenden Kapiteln soll die folgende Notation gelten, aus Gründen der Lesbarkeit werden die Operationen mit o, p, q bezeichnet:

$est(o)$ Frühest möglicher Startzeitpunkt von o ($e$arliest possible $s$tart $t$ime).

$ect(o)$ Frühest möglicher Endzeitpunkt von o ($e$arliest possible $c$ompletion $t$ime).

$lst(o)$ Spätest möglicher Startzeitpunkt von o ($l$atest possible $s$tart $t$ime).

$lct(o)$ Spätest möglicher Endzeitpunkt von o ($l$atest possible $c$ompletion $t$ime).

Das folgende Verfahren zur Sicherstellung von Kanten-Konsistenz basiert auf der Arbeit von Nuijten (1994) und wurde um reihenfolgeabhängige Übergänge erweitert. Zwischen zwei Operationen o, p sind reihenfolgeabhängige Übergangszeiten $t^{set}(o, p)$, $t^{set}(p, o)$ definiert. Die Operationen benötigen die gleiche Ressource $ra(o) = ra(p) = r$, deren Kapazität eins betragen soll, $cp(r) = 1$. Der Wertebereich D_o ist kanten-konsistent mit D_p, wenn Gleichung (4.16) erfüllt ist. Die beiden Operationen können in zwei unterschiedlichen Reihenfolgen eingeplant werden, $o \succ p$ oder $p \succ o$. Durch Untersuchen der beiden Reihenfolgen lassen sich Bedingungen zur Einschränkung der Wertebereiche ableiten. Die Reihenfolge $o \succ p$ ist gültig, wenn o zum Zeitpunkt $est(o) \leq t_s(o) \leq lst(p) - (pt(o, r) + t^{set}(o, p))$ oder früher eingeplant wird. Die Operation p kann auf jeden Fall an ihrem letztmöglichen Zeitpunkt $lst(p)$ begonnen werden. Hierbei ist die Übergangszeit zwischen o und p durch $t^{set}(o, p)$ berücksichtigt. Wird die Operation o nach p eingeplant, gilt also $p \succ o$, so kann dies konsistent zum Zeitpunkt $ect(p)+t^{set}(p, o) \leq t_s(o) \leq lst(o)$ oder später geschehen. Durch den frühest möglichen Fertigstellungstermin ist die Ausführung von p sichergestellt. Diese beiden Bedingungen werden zusammengefaßt. Es können alle Startzeitpunkte, die in diesem Intervall liegen, entfernt werden, da sie inkonsistent sind.

Die Kanten-Konsistenz von D_o zu D_p wird erreicht, wenn die folgenden Startzeitpunkte $t \in D_o$ entfernt werden:

$$lst(p) - (pt(o,r) + t^{set}(o,p)) < t < ect(p) + t^{set}(p,o) \tag{5.2}$$

Da die Konsistenz-Bedingung nur direktional gilt, muß (5.2) auch für D_p angewendet werden.

BEISPIEL: An Abbildung 5.2 wird die Sicherstellung der Kanten-Konsistenz verdeutlicht. Die beiden Operationen o, p seien von einer vorangegangenen, nicht betrachteten Stufe zugleich angekommen. Die Daten sind $pt(o) = 3$, $t^{set}(o,p) = 3$, $t^{due}(o) = 9$, $pt(p) = 4, t^{set}(p,o) = 1$, $t^{due}(o) = 13$, die Wertebereiche der Startzeitpunkte $D_o = \{0, \ldots, t^{due}(o) - pt(o,r)\} = \{0, \ldots, 6\}$, $D_p = \{0, \ldots, t^{due}(p) - pt(p,r)\} = \{0, \ldots, 9\}$. Hieraus lassen sich die Werte für die frühest möglichen Fertigstellungszeitpunkte und die spätest möglichen Starttermine bestimmen zu $ect(o) = 3$, $lst(o) = 6$, $ect(p) = 4$, $lst(p) = 9$. Durch Anwenden von (5.2) erhält man die konsistenten Wertebereiche

$$\begin{aligned} D_o &= \{0, \ldots, 6\} \backslash \{t | lst(p) - (pt(o,r) + t^{set}(o,p)) < t < ect(p) + t^{set}(o,p)\} \\ &= \{0, \ldots, 6\} \backslash \{4\} \\ D_p &= \{0, \ldots, 9\} \backslash \{t | lst(o) - (pt(p,r) + t^{set}(p,o)) < t < ect(o) + t^{set}(p,o)\} \\ &= \{0, \ldots, 9\} \backslash \{2, 3, 4, 5\} \end{aligned}$$

In Abbildung 5.3 sind die möglichen Belegungen dargestellt.

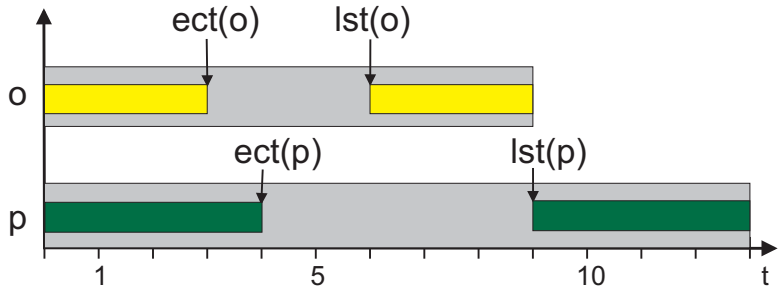


Abbildung 5.2.: Ausgangssituation zur Sicherstellung von Kanten-Konsistenz.

5.3.2. Ordering, Edge finding

Ist zwischen zwei Operationen o und p eine Reihenfolge (Ordering) $o \succ p$ definiert, so können die Wertebereiche evtl. eingeschränkt werden. Für die Startzeitpunkte

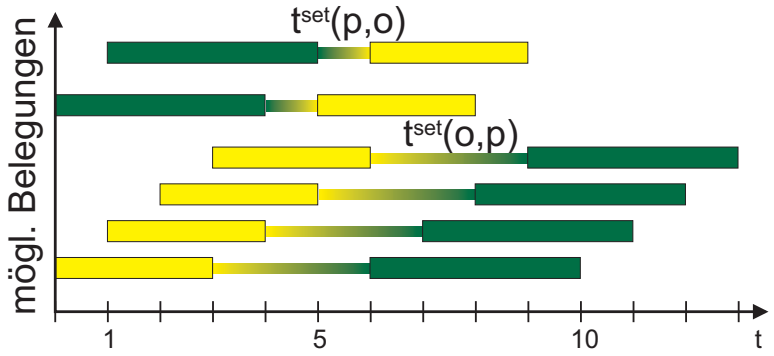


Abbildung 5.3.: Mögliche Belegung nach Entfernen der inkonsistenten Werte.

von Operation p und o muß gelten:

$$t_s(p) \geq ect(o) + t^{set}(o,p) \tag{5.3}$$
$$t_s(o) \leq lst(p) - t^{set}(o,p) \tag{5.4}$$

Somit lassen sich die Wertebereiche D_o und D_p einschränken zu

$$D_o \backslash \{t \in D_o | t > lst(p) - t^{set}(o,p)\}$$
$$D_p \backslash \{t \in D_p | t < ect(o) + t^{set}(o,p)\}$$

Die Anwendung auf das in Abbildung 5.2 dargestellte Problem zeigt Abbildung 5.4.

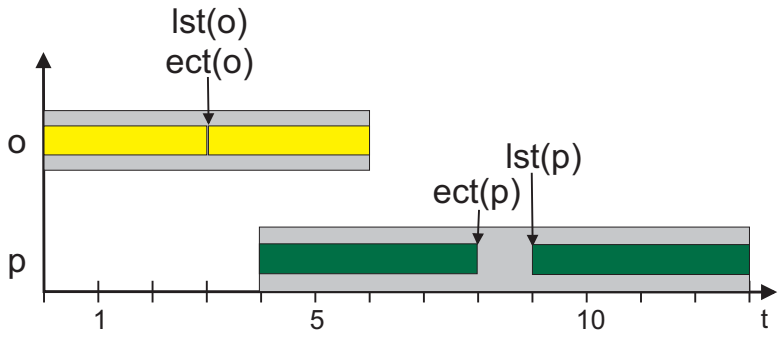


Abbildung 5.4.: Eingeschränkte Wertebereiche D_o und D_p, zugrunde liegt das in Abbildung 5.2 dargestellte Problem.

Bei dem sogenannten *edge-finding* liegt eine Menge von Operationen $S \subseteq \mathcal{O}$ vor. Es wird ermittelt, ob eine beliebige Operation $p \notin S$ in dem von S bestimmten

Intervall als erste oder letzte Operation eingeplant werden kann. Hierzu führen wir folgende Bezeichnungen ein:

$$\begin{aligned} est(S) &= min_{o \in S}\ est(o) \\ lct(S) &= max_{o \in S}\ lct(o) \\ pt(S) &= \sum_{o \in S} pt(o) \end{aligned}$$

Stellt p, $p \notin S$ die einzuplanende Operation dar und gilt

$$est(S \cup p) + pt(S \cup p) > lct(S) \tag{5.5}$$

so exisitiert kein Schedule, in dem p vor S eingeplant werden kann. Von Nuijten (1994) wird ein Fall aufgezeigt, in dem diese Bedingung nicht erfüllt ist, aber die untersuchte Operation nicht als erste eingeplant werden kann. Er erweitert die Bedingung, wenn für $p \notin S$

$$est(S) < est(p) < min_{r \in S}\ ect(r) \wedge pt(S \cup p) > lct(S) - est(p) \tag{5.6}$$

gilt, sind alle Zuweisungen $t \in D_p | t < min_{r \in S} ect(r)$ inkonsistent. Aus diesem Zusammenhang läßt sich eine untere $US(p)$ Schranke ableiten zu

$$\begin{aligned} &US(p) = max_{S \subset \mathcal{O}|\beta}\ min_{r \in S}\ ect(r) \\ &\beta \Leftrightarrow ra(S) = r \wedge p \notin S \wedge est(S) < est(p) < min_{r \in S}\ ect(r) \wedge \\ &pt(S \cup p) > lct(S) - est(p) \end{aligned} \tag{5.7}$$

Die obere Schranke $OS(p)$ ist entsprechend:

$$\begin{aligned} &OS(p) = min_{S \subset \mathcal{O}|\beta}\ max_{r \in S}\ lst(r) \\ &\beta \Leftrightarrow ra(S) = r \wedge p \notin S \wedge max_{r \in S} < lct(p) < lct(S) \wedge \\ &pt(S \cup p) > lct(p) - est(S) \end{aligned} \tag{5.8}$$

Der Beweis und eine Implementierung kann in Nuijten (1994) nachgelesen werden.

5.4. Sortierung von Variablen und Werten

Der ideale Suchprozeß wählt jede Variable einmal aus und weist ihr den optimalen Wert zu. Backtrack-Schritte sind nicht nötig, es werden ebenso viele Schritte benötigt, wie Variablen vorhanden sind. Komplexe Reihenfolgeprobleme, wie sie in dieser Arbeit untersucht werden, haben leider häufig Strukturen, die diesen idealen Suchprozeß nicht ermöglichen. Andererseits ist es einleuchtend, daß die Reihenfolge, in der die Variablen ausgewählt und eingeplant werden, einen entscheidenden Einfluß auf die Effizienz des Suchprozesses hat.

Den meisten Ansätzen zur Sortierung der Variablen liegt das *first-fail* Prinzip zugrunde. Dies besagt, daß schwierige Variablen, also solche, deren Instantiierung sehr wahrscheinlich zu einem Konflikt führt, zuerst ausgewählt werden sollten. Der Aufwand für die backtrack-Schritte kann somit reduziert werden, da Inkonsistenzen sehr früh festgestellt und somit auch umgangen werden können. Es wird in *statische* und *dynamische* Verfahren unterschieden. Bei statischen Verfahren wird die Reihenfolge der Variablen *a-priori* festgelegt und sowohl für eine Auswahl, als auch für die backtrack-Schritte verwendet. Der Rechenaufwand für die Sortier-Verfahren ist sehr gering, wird aber durch die ineffiziente Suche überkompensiert.

Bessere Resultate lassen sich mit dynamischen Sortierverfahren erzielen. Die Variablen werden abhängig von dem aktuellen Zustand des Suchprozesses ausgewählt. Nach einem backtrack-Schritt kann beispielsweise eine andere Reihenfolge von Variablen bestimmt werden. Der Rechenaufwand ist abhängig von dem implementierten Verfahren, allerdings kann auch der Suchprozeß effizienter sein.

Beispiele für dynamische Sortierverfahren sind:

- *Dynamic search arrangement - DSR*: An jedem Knoten des Suchbaumes wird die Variable ausgewählt, deren Wertebereich am kleinsten ist, MIN $|D|$.
- *Minimum width - MW*: Gegeben ist der Constraint-Graph, der die noch nicht instantiierten Variablen enthält. Die Variable (Knoten), von der die wenigsten Beziehungen (Kanten) ausgehen, wird als nächste ausgewählt.
- Ausnutzen von Eigenschaften der Operationen - sortiert wird nach dem spätest möglichen Endzeitpunkt $lct(o)$, spätest möglichen Startzeitpunkt $lst(o)$ oder frühest möglichen Startzeitpunkt $est(o)$ einer Operation.
- Kombinations-Verfahren: Bilden der Menge von Variablen, die noch nicht ausgewählt wurde. Aus dieser Menge wird die Untermenge der Variablen gebildet, die als erste in dem Schedule eingeplant werden können (durch

Reihenfolge-Constraints eingeschränkt). Aus dieser Menge wird die Variable ausgewählt, die den kleinsten/größten Wertebereich oder den kleinsten/größten Fertigstellungszeitpunkt hat. Hier sind zahlreiche Kombinationen möglich. In den Kapiteln 6 und 7 werden diese Verfahren und das letztgenannte verwendet.

Von Sadeh (1991) wird anhand von Beispielen aus dem Bereich des Scheduling gezeigt, daß diese Heuristiken deutliche Schwächen aufweisen. Bei der Einplanung von Operationen auf Ressourcen ist die Beachtung der Beschränktheit (engl. *tightness*) einer Ressource, einer Operation oder eines abstrakten Gebildes wie eines Constraints sehr wichtig.

In den Arbeiten von Sadeh (1991), Sadeh (1994) und Sadeh und Fox (1996) werden Sortierungsverfahren entwickelt, um die Anzahl der backtrack-Schritte zu minimieren. Die zugrunde liegende Idee ist es, die *kritischen* Variablen, die sehr wahrscheinlich zu einem backtrack-Schritt führen, zuerst einzuplanen und zwar mit einem *guten* Wert, der in einer großen Anzahl an Lösungen enthalten ist. Sie definieren die Wahrscheinlichkeit, mit der eine Ressource zu einem bestimmten Zeitpunkt von einer Operation belegt ist. Dieser Wert wird als *Bedarf* b bezeichnet und folgendermaßen berechnet:

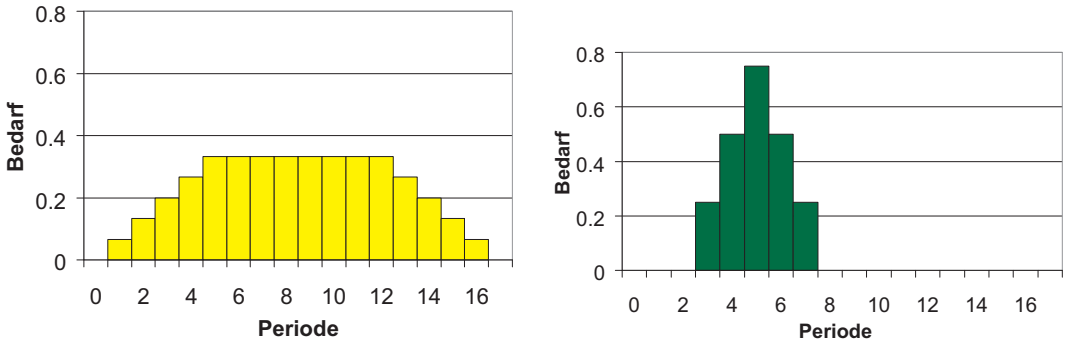


Abbildung 5.5.: Bedarfs-Profile für die Operationen o mit $pt(o) = 5$, $D_o = \{1, \ldots, 11\}$ (links) sowie Operation p mit $pt(p) = 3$, $D_p = \{3, 4\}$.

Der Wertebereich der Startzeitpunkte der Operation o habe die Größe $d = |D_o|$, die Bearbeitungszeit sei $pt(o)$. Die Wahrscheinlichkeit, daß die Operation zu einem beliebigen Zeitpunkt $t \in D_o$ startet, ist $\frac{1}{d}$. Die Wahrscheinlichkeit, daß die Ressource von $t, \ldots, t + pt(o) - 1$ von Operation o belegt ist, beträgt ebenfalls $\frac{1}{d}$. Durch Aufsummieren der Wahrscheinlichkeiten für alle möglichen Startzeitpunkte erhält man eine Verteilung der Wahrscheinlichkeit der Belegung oder (nach Sadeh) ein *Bedarfs-Profil*. In Abbildung 5.5 sind Bedarfs-Profile für eine Operation

mit unterschiedlichen Wertebereichen und für zwei um eine Ressource konkurrierende Operationen angegeben. Der Bedarf einer Operation o zu einem beliebigen Zeitpunkt t sei mit $b(o,t)$ bezeichnet.

Die Auswahl der Variablen richtet sich nach den erzeugten Bedarfs-Profilen. Für jede Ressource wird die Bedarfs-Spitze bestimmt und anschließend die Operation eingeplant, deren Bedarf in dieser Spitze am höchsten ist.

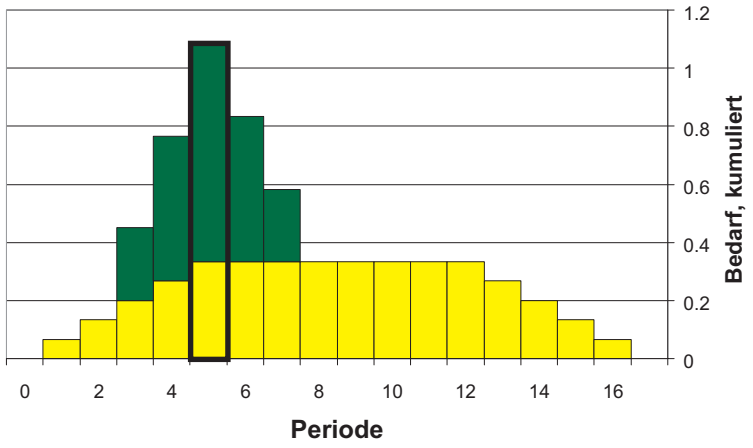


Abbildung 5.6.: Bedarfs-Profil einer Ressource r, auf der die beiden Operationen o und p eingeplant wurden.

Von Breitinger und Lock (1994) wird kritisiert, daß die Bedarfs-Profile bei jeder Änderung, also Instantiierung einer Variablen, neu berechnet werden müssen, was einen erheblichen Zusatzaufwand bei der Suche bedeutet. Auch andere Heuristiken von Sadeh (1991) sind durch die erneute Berechnung nach einer Änderung sehr zeitintensiv. Breitinger und Lock (1994) führen *semi-dynamische* Selektionskriterien ein. Hier wird der Untersuchungszeitraum in k gleich große Bereiche $A_1, \ldots, A_k$ mit der Größe $s_i = |A_i|$ eingeteilt. Der Bedarf der einzelnen Operationen in diesen Bereichen wird summiert und durch die Größe des Bereiches geteilt

$$b_{rel}(o, A_i) = \frac{\sum_{t \in A_i} b(o,t)}{|A_i|} \tag{5.9}$$

Weiterhin wird bestimmt, wie stark sich die Wertebereiche der Variablen mit den Bereichen überdecken. Es ergibt sich

$$f(o, A_i) = \frac{|A_i \cap D_o|}{|D_o|} \tag{5.10}$$

Der Bedarf berechnet sich nun zu

$$b_{neu}(o, A_i) = b_{rel}(o, A_i) \cdot f(o, A_i) \tag{5.11}$$

Die Operationen werden nach $b_{neu}(o, A_i)$ priorisiert. Der Vorteil dieses Verfahrens besteht nach Meinung der Autoren darin, daß die Bedarfswerte nur zu Beginn berechnet werden müssen. Allerdings werden die Operationen hier nicht zu ihrem frühest möglichen Zeitpunkt eingeplant, sondern in dem Teilbereich, der am stärksten mit dem Wertebereich überlappt und in dem die Belegung der Ressource mit der Operation am höchsten ist. Bei Problemen, deren Startzeitpunkte nicht gleich sind, kann es somit zu Lücken in dem erzeugten Schedule kommen, was bei einer Anwendung, bei der es auf die Einhaltung von Fertigstellungszeitpunkten ankommt, zu vermeiden ist. Von den Autoren werden nur Anwendungen mit dem Ziel der Minimierung der Durchlaufzeit angegeben.

5.5. Constraint Optimization Problem (COP)

Durch Lösen des CSP wird eine gültige Lösung des Problems bestimmt. Eine Aussage über die Güte einer Lösung oder die Vergleichbarkeit mit anderen gültigen Lösungen wird über die Bewertung der Lösung möglich. Hierzu wird eine Zielfunktion $f(x)$ definiert, die zu minimieren, bzw. zu maximieren ist.

Ist S_0 die Lösung des Problems P_0, können wir ein neues Problem P_1 definieren, dessen Lösung besser als S_0 sein soll. Dies wird solange wiederholt, bis Problem P_{n+1} nicht mehr lösbar ist. Die Lösung S_n zu Problem P_n ist also die optimale Lösung, siehe Le Pape (1994).

Diese schrittweise Verbesserung kann durch Hinzufügen von $c_n : f(x) < y$ (bei zu minimierendem $f(x)$) zu der Menge von Constraints realisiert werden. Das entstehende Problem wird als *Constraint Optimization Problem (COP)* bezeichnet. Kann das CSP gelöst werden, wird c_n mit einem kleineren y zu der Menge der Constraints hinzugefügt und das CSP erneut gelöst. Somit ist sichergestellt, daß keine Lösung verlorengeht. Dieser Vorgang wird solange wiederholt, bis das COP nicht mehr lösbar ist, der Zielfunktionswert der letzten gültigen Lösung entspricht dem Optimum.

In den folgenden beiden Kapiteln wurde diese Eigenschaft genutzt, um die Optimalität einer Lösung zu zeigen. Die in den Anwendungskapiteln verwendeten (zu minimierenden) Zielfunktionen sind wie die Variablen auf finiten Wertebereichen definiert, wenn also das CSP für den Zielfunktionswert y_1 lösbar und für $y_1 - 1$ nicht lösbar ist, stellt y_1 die optimale Lösung dar. Heuristiken, die Eigenschaften der Zielfunktion nutzen, können in lokalen Minima festhängen; ein ähnliches

Verhalten wie das bei CSP bekannte trashing kann beobachtet werden. Dieses Verhalten tritt bei einem COP nicht auf, da das erzeugte CSP lediglich lösbar bzw. nicht lösbar sein kann. Die Struktur des Suchraumes kann allerdings bewirken, das bei einem lokalen Minimum das Bestimmen der Lösung des CSP deutlich mehr Zeit in Anspruch nimmt, als bei einer schlechteren Lösung.

Die gefundene (gültige) Lösung sollte ebenfalls zur Einschränkung des Suchraumes verwendet werden. Wird das Problem durch Hinzufügen des beschriebenen Constraints $c_n : f(x) < y$ eingeschränkt und weist die gefunden Lösung $f(x')$ einen kleineren Wert $y' < y$ auf, so ist es sinnlos, den *Slack*-Bereich $\Delta_s = y - y'$ zu untersuchen. Somit wird im nächsten Schritt y' als obere Schranke verwendet und das Constraint $c_n : f(x) < y' - 1$ hinzugefügt.

Es ist einleuchtend, daß dieser Suchprozeß sehr rechenintensiv sein kann, insbesondere, wenn der Wert y noch weit von dem optimalen Wert MIN $f(x)$ entfernt ist. Effizientere Suchverfahren basieren auf einer größeren Schrittweite bei Einschränkung des Suchraumes, ein Beispiel ist die *binäre Suche*. Ist $f(x) = Y$ ein bekannter Lösungswert eines Minimierungsproblems, wird der zu untersuchende Bereich über $[a, b]$, mit $a = 0$ und $b = Y$, definiert. Nun wird versucht, das CSP mit $f(x) < \frac{a+b}{2}$ zu lösen. Ist das CSP lösbar, d.h. kann ein $f(x)$ mit $f(x) < \frac{a+b}{2}$ gefunden werden, wird die obere Schranke zu $b = \frac{a+b}{2}$ angepaßt und die Suche beginnt erneut. Konnte keine Lösung gefunden werden, wird die untere Schranke zu $a = \frac{a+b}{2}$ angepaßt und erneut gesucht. Durch die Verwendung von sehr guten unteren und oberen Schranken zur initialisierung von a und b wird die Suche effizienter.

Andere Verfahren verwenden eine andere Aufteilung des Suchraumes, z.B. wird der Bereich nicht in $[a, \frac{a+b}{2}), [\frac{a+b}{2}, b]$, sondern in $[a, \frac{a+b}{3}), [\frac{a+b}{3}, b]$ aufgeteilt.

Leider haben sehr schwere Probleme ein ausgeprägtes *Phasen-Übergangs*-Verhalten, d.h. die Bestimmung von Lösungen im Bereich des Optimum ist äußerst rechenaufwendig im Vergleich zu schlechten (Rand-)Lösungen. Hier gestaltet sich die Bestimmung der besten Suchstrategie oder der Schrittweite bei der Suche äußerst schwierig.

Die Zielfunktion $f(x)$ besteht aus einer (häufig additiven) Kombination von Unterzielen $h_1(x), h_2(x), \ldots, h_n(x)$, für die Präferenzen $g_1, g_2, \ldots, g_n$ (häufig auch als Gewichte bezeichnet) angegeben werden $f(x) = g_1 \cdot h_1(x) + \ldots + g_n \cdot h_n(x)$. Bei einem Umschlagsterminal sind dies beispielsweise die Auslastung der Ressourcen, die Einhaltung des Fahrplanes, die Minimierung der Umschläge, etc. Jedes Unterziel wird durch die Summe der Ausprägungen der einzelnen Vorgänge bestimmt, stellt somit wieder eine aggregierte Bewertung dar. In der Entscheidungstheorie, beispielsweise bei Eisenführ und Weber (1993), werden unterschiedliche Verfah-

ren vorgeschlagen, um die Gewichte zu bestimmen. Über die Gewichtung wird versucht, eine Priorisierung der Unterziele vorzunehmen. Um Dominanzeffekte zu verhindern, muß die Gestalt der Zielfunktion bekannt sein. Werden sprungfixe Unterziele, wie beispielsweise die Minimierung der maximalen Verspätung, bezeichnet mit L, von engl. *lateness* MIN $f(x)$ = MIN $(maxL_i)$ verwendet, ist die Gestalt und damit die korrekte Priorisierung sehr schwer faßbar. Relaxierbare Constraints werden häufig in Form von *Straftermen* in die Zielfunktion aufgenommen, was die Komplexität der Zielfunktion erhöht und damit die objektive Bewertung der Lösung erschwert.

Die Unterziele werden bei einem Constraint Optimization Problem durch eine Menge von Constraints realisiert. Mit einer Priorisierung der Constraints und der Vorgabe von möglichen Relaxierungen der Constraints läßt sich der Suchprozeß wesentlich genauer steuern. Das Konzept der Suche ist gleich, nur wird der Schritt der Transformation der Unterziele in die Zielfunktion vermieden, wodurch eine wesentlich natürlichere Steuerung des Suchprozesses möglich ist.

5.6. Zusammenfassung

Die modellierten mehrstufigen Umschlagsysteme werden als ein Constraint Satisfaction Problem (CSP) bzw. bei Optimierungsfragestellungen als ein Constraint Optimization Problem (COP) formuliert. Dieser Schritt überträgt das verfahrensunabhängige Modell auf eine verfahrensabhängige Formulierung. In diesem Kapitel wurden Variablen, Wertebereiche und Constraints eingeführt, um ein Constraint Satisfaction Problem zu definieren. Anschließend wurden unterschiedliche Konzepte zur Lösung dieses Problems vorgestellt. Backtracking, Konsistenz-Prüfung und Sicherstellung, die Bestimmung von impliziten Reihenfolgen und die Sortierung von Variablen und Werten tragen zu einer effizienten Suche bei. Es wurde beschrieben, wie die Ansätze auf mehrstufige Umschlagsysteme angewendet werden können. Wird eine Zielfunktion minimiert/maximiert, kann auf Grundlage eines CSP ein COP formuliert werden.

In den folgenden Kapiteln wird nun die Modellierung, Formulierung und Implementierung auf zwei praktische Fälle angewendet.

6. Der COP-Ansatz für mehrstufige Kommissioniersysteme

Die Kunden werden anspruchsvoller, es wird in immer kleineren Mengen bestellt, und ein 24h-Lieferservice ist in vielen Bereichen inzwischen Standard. Die Einsparungspotentiale im Bereich der Logistik sind immer noch gewaltig, mit der Folge, daß in den Distributionszentren neue Anforderungen an die komplexen Kommissioniersysteme gestellt werden. Kurze Durchlaufzeiten und hohe Flexibilität sind die Ziele. In realen Anlagen findet man oft unzureichend dimensionierte Puffer, Seiteneffekte durch Priorisierung von Eilaufträgen, Leerlaufzeiten durch die Bildung von Warteschlangen, eine schwierige Personaleinsatzplanung und schwer prognostizierbare und damit schwer einhaltbare Fertigstellungstermine. Diese Ziele (hoher Servicegrad, niedrige Bestände und hohe Flexibilität) können auf Basis eines durchgängigen Informationsflusses mit Hilfe von Optimierungsverfahren erreicht werden.

Wie bereits zitiert hat nach (VDI 3590) Kommissionieren das Ziel, „aus einer Gesamtmenge von Gütern (Sortiment) Teilmengen auf Grund von Anforderungen (Aufträge) zusammenzustellen“. Hierbei besteht ein Auftrag aus n Positionen. Jede Position entspricht genau einem Artikel in einer definierten Menge. Für Aufträge gibt es in der Regel Wunschtermine (spätest mögliche Fertigstellungszeitpunkte), die beispielsweise von den Abfahrtzeiten der Touren der ausliefernden Dienstleister bestimmt werden. Wird der Termin überschritten, können die betreffenden Artikel erst in der nächsten Tour ausgeliefert werden. Der Servicegrad, der über die Anzahl pünktlich ausgelieferter Aufträge definiert ist, verschlechtert sich. Das Ziel eines effizienten Kommissioniersystems ist es daher, die vorgegebenen Fertigstellungszeitpunkte bei minimalem Aufwand einzuhalten.

In Kapitel 6.1 wird das untersuchte System beschrieben. Die Modellierung schließt sich in Kapitel 6.2 an. Es wird unterschieden, ob die Zusammenfassung (Clusterung) der Aufträge zu Batches abgeschlossen ist oder ob sie mit in die Modellierung aufgenommen wird. Für diese beiden Fälle wird eine Modellierung entwickelt. In Kapitel 6.3 wird die Modellierung genutzt, um ein Constraint Satis-

faction Problem zu definieren. In Kapitel 6.4 wird die Erweiterung des CSP als Constraint Optimization Problem vorgestellt. In Kapitel 6.5 wird das COP auf unterschiedliche Probleme angewendet, die Ergebnisse werden diskutiert und bewertet.

6.1. Mehrstufige Kommissioniersysteme

Ein Haupt-Unterscheidungskriterium bei Kommissioniersystemen ist die realisierte Bereitstellung der Ware, die *statisch* oder *dynamisch* sein kann. Bei der statischen Bereitstellung liegen die Artikel in Ladehilfsmitteln in Fächern, der Kommissionierer bewegt sich zu den Fächern. Diese Variante wird umgangssprachlich „Mann-zur-Ware“ genannt. Bei der dynamischen Bereitstellung, auch als „Ware-zum-Mann“ bezeichnet, werden die Artikel über automatische Fördermittel aus dem Lager zu dem Kommissionierplatz gefördert. Nach der Entnahme werden die Restmengen der Artikel wieder in das Lager zurückgegeben. In der Regel findet bei der Kommissionierung ein Übergang zu einem kleineren Ladehilfsmittel statt, beispielsweise von Europaletten zu Behältern. Für eine detaillierte Einführung in Kommissioniersysteme sei auf Arnold (1995), Gudehus (1973) und Jünemann (1989) verwiesen.

Eine Kommissionierung nach dem dynamischen Bereitstellprinzip wird häufig mehrstufig betrieben. In der ersten Stufe wird *artikelorientiert* kommissioniert, in der zweiten *auftragsorientiert* verpackt. In einer weiteren Stufe können beispielsweise unterschiedliche Touren zusammengestellt werden. Auf jeder Stufe stehen parallele Arbeitsplätze zur Verfügung, siehe Abbildung 6.1. Die einzelnen Stufen sind durch Puffer entkoppelt, hier können die Produkte beliebig umsortiert werden. In der Regel sind die Puffer automatische Behälterlager oder Systeme von Rollenbahnen, auf denen zusätzlich noch eine Sortierfunktion ausgeführt wird.

Die Aufträge können je nach Branche unterschiedlich sein. Im reinen Ersatzteilgeschäft wird beispielsweise unterschieden in

- *Lageraufträge,* um die dezentralen Bestände der Großhändler aufzufüllen. Diese Aufträge haben große Stückzahlen pro Position. Der Zeitpunkt des Auftragseinganges und der gewünschte Auslieferungstermin liegen weit auseinander.

- *Eilaufträge,* z.B. zur Instandsetzung einer ausgefallenen Maschine. Diese Aufträge bestehen aus wenigen Positionen mit kleinen Stückzahlen (oft Stückzahl eins). Eilaufträge erfordern eine schnellst mögliche Reaktion.

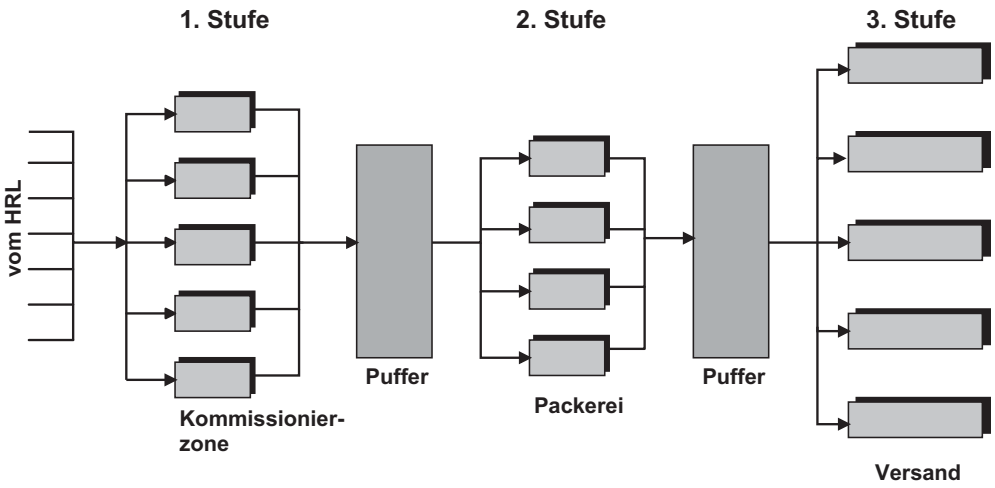


Abbildung 6.1.: Mehrstufiges Kommissioniersystem, auf jeder Stufe befinden sich parallele Bearbeitungsplätze, die Stufen sind durch Puffer voneinander entkoppelt.

Bei Handelsunternehmen oder Versandhäusern werden die Aufträge nicht unterschieden, allerdings wird hier durch das Bestellverhalten eine gewisse Inhomogenität erzeugt.

Der Engpaß eines Systems sollte durch eine Pull-Steuerung optimal ausgelastet werden, um einen maximalen Durchsatz zu gewährleisten, siehe Goldratt und Cox (1998). Jeder Bereich eines mehrstufigen Systems kann (temporär) der Engpaß des Systems sein, siehe Alicke und Faisst (1999).

6.2. Modellierung

Die Ressourcen des Systems sind die Regalbediengeräte (RBG) zur Auslagerung der Artikel, die Vorzone mit den Fördermitteln, die Kommissionierplätze, die Packplätze, das benötigte Personal und die Puffer. Operationen sind das Auslagern der Artikel, Tätigkeiten wie das Kommissionieren eines Artikels in einer gegebenen Menge, das Verpacken mehrerer Artikel und die Bildung von Versandeinheiten. Operationen sind an Ressourcen gebunden. Die Operationen zwischen der Stufe Kommissionieren und Verpacken sind durch eine Mindesttransportzeit voneinander getrennt, ebenso die Stufen Verpacken und Versenden (oder versandfertig machen). Die parallelen Arbeitsplätze stellen parallele Ressourcen oder Ressourcen mit Kapazität $cp > 1$ dar. Die Leistungsfähigkeit der einzelnen Plätze sei gleich. Bei dem untersuchten System ist nicht das Transportsystem der Engpaß des Systems. Dies ist aus planerischen Gesichtspunkten auch sinnvoll, da

Kommissionier- und Packplätze in einem gewissen Umfang hinzugefügt werden können, die Transportkapazität allerdings durch den Grenzdurchsatz des Engpaßelements determiniert ist.

Das System kann als mehrstufiges Umschlagsystem (mit begrenztem Puffer) modelliert werden. Erschwerend kommt hinzu, daß die Entscheidung, wie viele Positionen eines Artikels in der ersten Stufe auf einmal kommissioniert werden, gleichzeitig mit der Reihenfolgeentscheidung getroffen werden kann. Übertragen auf Job-Shop Probleme hieße dies, die Losgröße simultan zu der Reihenfolge zu bestimmen. In der Regel werden solche Systeme über eine Batch-Bildung gesteuert, d.h. die Aufträge für eine bestimmte Zeit Δt oder eine gegebene Anzahl von Aufträgen werden zusammengefaßt und abgearbeitet. Danach wird der nächste Batch angefangen und abgearbeitet. In der vorliegenden Arbeit sollen zwei unterschiedliche Fälle untersucht werden:

1. In einem Batch wird jeder Artikel nur einmal ausgelagert, alle Aufträge, die diesen Artikel enthalten, werden auf einmal befriedigt (Fall 1 - Beachtung der Batch-Zuweisung).

2. Die Artikel, die in einem Batch zu kommissionieren sind, können mehrfach ausgelagert werden, bis hin zur auftragsorientierten Kommissionierung (Fall 2 - Keine Beachtung der Batch-Zuweisung).

Die Auswirkungen der Batchbildung ist in Abbildung 6.2 dargestellt. Auf jeder Stufe ist eine Ressource vorhanden, für das Hochregallager ist nur die Auslagerung dargestellt. Jeder Auftrag besteht aus den vier Artikeln 1, 2, 3 und 4. Auftrag A und B bilden Tour T1, Auftrag C bildet Tour T2 und Auftrag D die Tour T3. Die Touren sind im Vorhinein bekannt, z.B. durch Gebiete, die von externen Dienstleistern bedient werden. Die Struktur des Beispiels ist in Abbildung 6.3 dargestellt.

In dem oberen Diagramm (Abbildung 6.2) ist der Extremfall nach Fall 2 dargestellt, die auftragsorientierte Kommissionierung. Jeder zu kommissionierende Artikel wird ausgelagert, zu den Kommissionierplätzen transportiert und bearbeitet, anschließend wird die Restmenge wieder zurückgelagert. Nachdem jeder Artikel einmal kommissioniert und in den Puffer transportiert wurde, kann mit dem Verpacken begonnen werden. Anschließend ist das Packstück versandbereit. In dem unteren Gantt-Diagramm ist die Kommissionierung nach Fall 1 dargestellt. Hier werden alle Aufträge eines Batches, die einen Artikel enthalten, auf einmal befriedigt. Die Artikel werden gepuffert, sobald ein Auftrag komplett ist, verpackt und anschließend versendet. Anhand der Abbildung können die folgenden Aussagen gemacht werden:

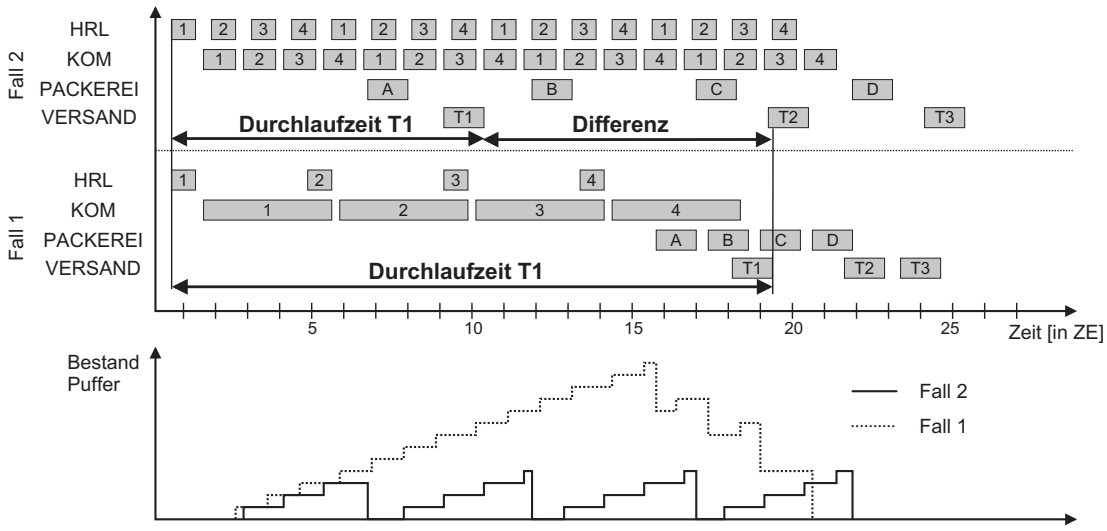


Abbildung 6.2.: Darstellung des Gantt-Diagramms für unterschiedliche Kombinationen von Aufträgen, Fall 1 - Beachtung der Batch-Zuweisung, Fall 2 - keine Beachtung der Batchzuweisung. Hier dargestellt sind die beiden Extremfälle: Fall 1 - artikelorientierte Kommissioierung und Fall 2 - auftragsorientierte Kommissionierung

- **Durchlaufzeit eines Auftrages**: Die Durchlaufzeit ist bei Fall 2 deutlich geringer, in Abbildung 6.2 als Differenz dargestellt, da direkt nach der Kommissionierung verpackt und versandt werden kann. In Fall 1 müssen alle Positionen, die einen Artikel enthalten, nacheinander kommissioniert werden. Der Artikel kann nicht in das Hochregallager (HRL) zurückgelagert werden. Ist ein Fertigstellungszeitpunkt von $t^{due}(T1) = 11[ZE]$ gegeben, kann dieser nur bei Fall 2 erreicht werden.

- **Gesamt-Durchlaufzeit aller Aufträge**: Durch die Übergangszeiten an den Stufen (hier an den Kommissionierplätzen) ist die gesamte Durchlaufzeit bei Fall 2 größer als bei Fall 1.

- **Bestand des Puffers**: In Fall 2 muß immer nur ein Auftrag gepuffert werden, somit entspricht der maximale Bestand des Puffers der maximalen Anzahl Positionen eines Auftrages. Eine deutlich stärkere Belastung des Puffers ergibt sich bei Fall 1. Hier werden alle Artikel gepuffert, bis der erste Auftrag komplett ist und zur Packerei transportiert werden kann.

- **Auslastung der einzelnen Stufen**: Die Auslastung des HRL ist bei Fall 2 sehr hoch. Bei Fall 1 konzentrieren sich die Aufträge für den Pack- und

Versandbereich am Ende, bei Fall 2 sind sie besser über den Zeitraum verteilt.

Die Aussagen sollen deutlich machen, daß die Integration aller (bearbeitenden und puffernden) Ressourcen für die Modellierung und Optimierung des Gesamtsystems äußerst wichtig ist.

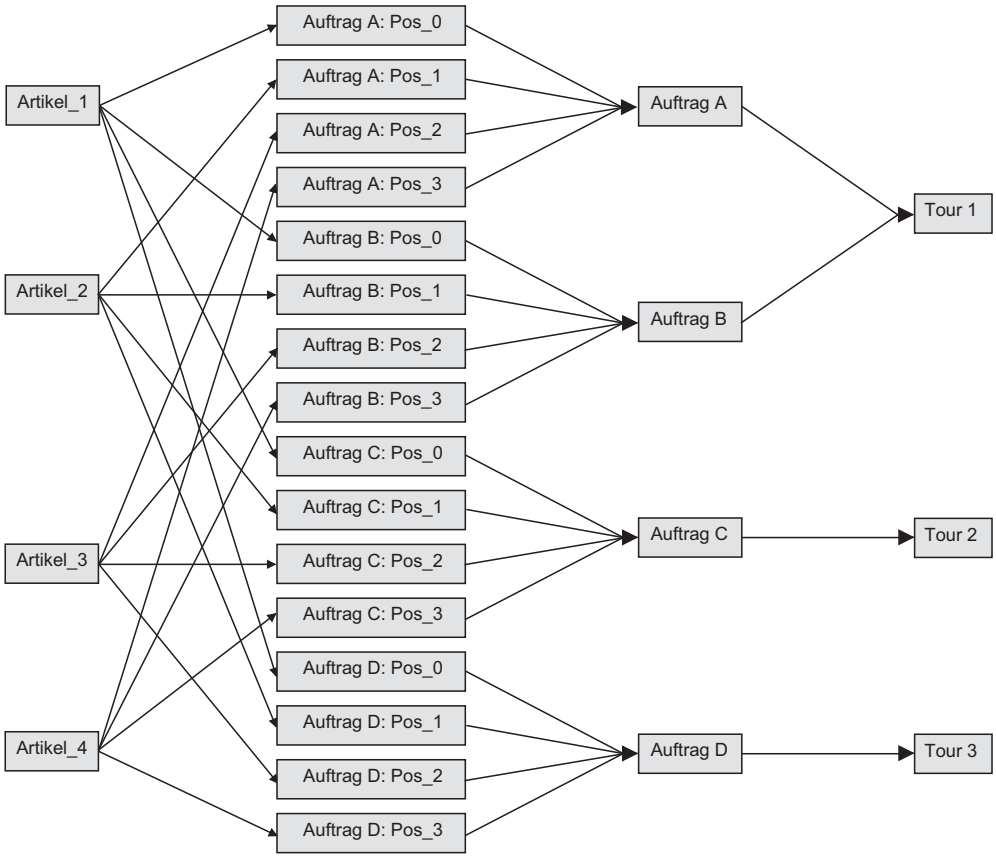


Abbildung 6.3.: Struktur des Beispiels, jeder Auftrag besteht aus vier Positionen, die jeweils einem Artikel entsprechen.

Der Eingang von Aufträgen ist in der Regel nicht kontinuierlich, sondern es lassen sich deutliche Spitzen erkennen, was auf *außerbetriebliche* Einflußgrößen zurückzuführen sein kann. Diese Spitzen entstehen, wenn beispielsweise ein Annahmeschluß für Bestellungen, die noch am selben Tag verschickt werden, vorgegeben ist. Verdeutlicht man sich den Verlauf über die einzelnen Stufen hinweg, ergibt sich eine Wanderung der Belastungsspitzen, die in Abbildung 6.4 dargestellt ist. Es kann zwischen *langfristigen* und *kurzfristigen* Engpässen des Systems unterschieden werden. Der langfristige Engpaß ist systemimmanent und wird durch

die *Hardware*, also beispielweise die Anzahl der Kommissionierplätze oder RBG, bestimmt. Mit dem Einsatz von flexiblen Ressourcen (Personal, Stapler, mobile Kommissionierplätze etc.) wird die kurzfristige Leistungsfähigkeit des Systems bestimmt. Beispielsweise können bei niedrigem Auftragseingang nur die Hälfte der Kommissionier- und Packplätze genutzt werden. Hieraus ergibt sich ein kurzfristiger Engpaß, der durch den Einsatz der flexiblen Ressourcen bestimmt wird. Auf diese Weise können die beschriebenen Belastungsspitzen ausgeglichen werden. Die flexiblen Ressourcen werden als Pool modelliert (siehe Abbildung 6.5).

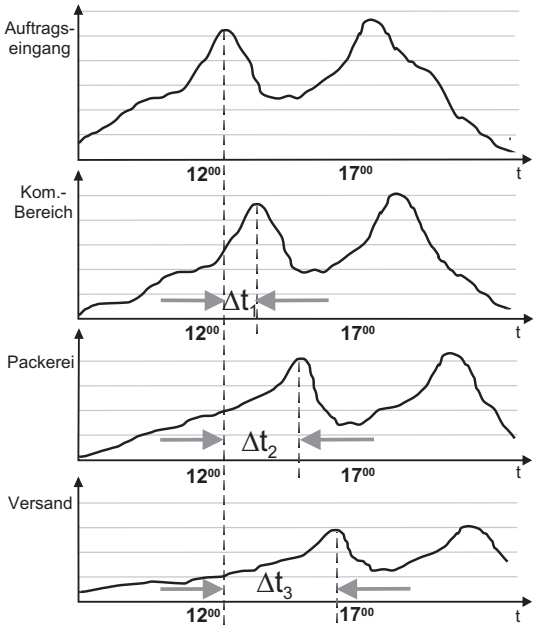


Abbildung 6.4.: Spitzen, die durch ein mehrstufiges Kommissioniersystem *wandern*.

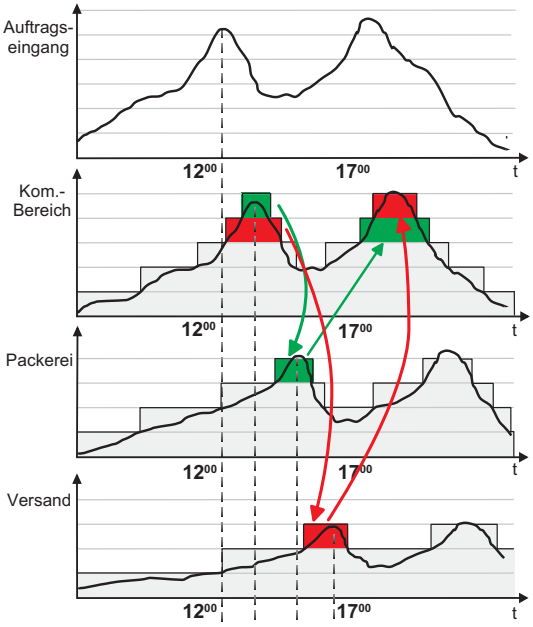


Abbildung 6.5.: Ausgleich durch Beachtung eines kapazitativ modellierten Pools von flexiblen Ressourcen.

Die Ansätze zur Glättung des Einsatzes von flexiblen Ressourcen, beispielsweise durch ein vorausschauendes Kommissionieren auf Lager, ist aufgrund der kurzen Reaktionszeit nicht möglich. Mitarbeiter sollten flexibel, beispielsweise über variable Schichtzeiten, Kurzarbeit und wechselnde Einsatzorte (Kommissionierbereich, Packerei, Versand) eingesetzt werden (siehe Günther (1989), Günther und Tempelmeier (2000)). Um einen solchen flexiblen Einsatz zu planen, bedarf es eines integrierten Modells, das alle Abhängigkeiten in einem angemessenen Aggregationsgrad berücksichtigt.

Zusätzlich zu dem Auftrag werden die Begriffe der *Position* und der *Tour* eingeführt, um die Struktur zu modellieren. Es handelt sich bis zu den Kommis-

sionierplätzen um eine *out-tree*-, nach der Kommissionierstufe um eine *in-tree*-Struktur. In Abbildung 6.6 ist diese Struktur zusammen mit der Modellierung nach Rumbaugh, Blaha und Premerlani (1993) dargestellt. Anhand des Objektmodells lassen sich die folgenden Definitionen einfacher verstehen.

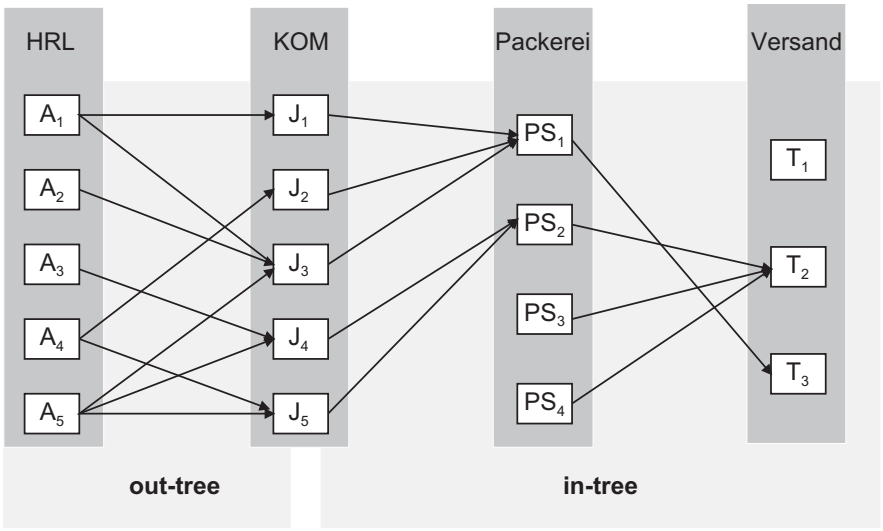


Abbildung 6.6.: Produkt-Struktur eines mehrstufigen Kommissioniersystems, bis zu den Kommissionierplätzen als out-tree, danach als in-tree.

Zugrunde liegt eine Menge von Artikeln $a \in \mathcal{A}$. Jeder Auftrag $j \in \mathcal{J}$ besteht aus einer Menge von Positionen $pos \in \mathcal{POS}$, $\mathcal{J} = P(\mathcal{POS})$. Eine Position eines Auftrages ist immer genau einem Auftrag und Artikel zugeordnet. Die Größe der Position, und hiervon abgeleitet die Bearbeitungszeit an den Kommissionierplätzen, ist gegeben zu $q : \mathcal{POS} \to \mathbb{N}^+$. Der Artikel der Position ist $as : \mathcal{POS} \to \mathcal{A}$. Somit besteht ein Auftrag aus einer Menge von Artikeln, was durch $aj : \mathcal{J} \to \mathcal{A}$ gegeben ist, die enthaltenen Positionen werden durch $ap : \mathcal{J} \to \mathcal{POS}$ definiert. Mehrere Aufträge werden zu einer Tour $t \in \mathcal{T}$ zusammengefaßt, $\mathcal{T} = P(\mathcal{J})$. Die enthaltenen Aufträge sind $at : \mathcal{T} \to \mathcal{J}$. Für jede Tour sind Fertigstellungszeitpunkte $t^{due} : \mathcal{T} \to \mathbb{N}^+$ und für die Aufträge Startzeitpunkte (Auftragseingang in das System) $t^{rel} : \mathcal{T} \to \mathbb{N}^+$ gegeben. Eine Tour kann beispielsweise durch den Zeitplan eines Spediteurs gegeben sein.

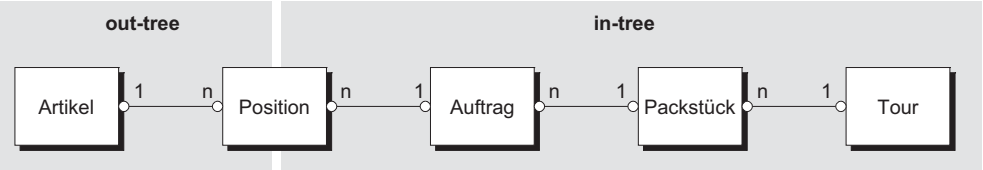


Abbildung 6.7.: Objekt-Struktur eines mehrstufigen Kommissioniersystems.

6.3. Modellierung und Formulierung als Constraint Optimization Problem

Die in Kapitel 4 vorgestellten Constraints werden genutzt, um die beiden unterschiedlichen Fälle als COP zu modellieren und formulieren.

Die Bearbeitungsplätze der Stufen werden zu je einer parallelen Ressource zusammengefaßt. So bezeichnet g die Auslagergassen, k die Kommissionierung, p die Packerei und v den Versand. b_{kp} und b_{pv} stellen die Puffer zwischen Kommissionierung und Packerei bzw. zwischen Packerei und Versand dar. Mit h wird der Personalpool bezeichnet. Die Menge aller Ressourcen ist $\mathcal{R} = \{g, k, p, v, b_{kp}, b_{pv}, h\}$. Die Anzahl parallel arbeitender Plätze ist über die Kapazität der Ressourcen gegeben. Beispielsweise ist bei vier parallelen Kommissionierplätzen $cp(k) = 4$. Das Transportsystem wird nicht als beschränkte Ressource angenommen, hier wird lediglich die Transportzeit berücksichtigt. Zur Ausführung einer Operation wird eine Ressource k, p oder v und zusätzlich der Personal-Pool h benötigt. Der Bedarf zur Bearbeitung eines Auftrages, Packstückes oder Tour ist $sz(o, r) = 1 \ \forall \ o, r = \{k, p, v\}$. Der zusätzliche Personalbedarf ist für jede Ressource ebenfalls $sz(o, h) = 1, \ \forall \ o$. Die von dem Personal-Pool bedienten Ressourcen werden zusammengefaßt zu $rs_p = \{k, p, v\}$.

6.3.1. Fall 1 - Beachtung der Batch-Zuweisung

O^a_{hrl} sei die Operation, den Artikel a auszulagern. Alle Aufträge mit diesem Artikel werden auf einmal befriedigt. Die Operationen zur Modellierung des Kommissioniervorganges seien O^{pos}, die Dauer der Operation $pt(O^{pos})$ hängt von der zu kommissionierenden Menge $q(pos)$ ab. Die Kommissionierung kann begonnen werden, wenn die Auslagerung und der Transport zu den Kommissionierplätzen abgeschlossen ist, also muß mit der Zeit $t_{g,k}$ für den Transport von den Regalgassen zu den Kommmissionierplätzen gelten

$$t_e(O^a_{hrl}) + t_{g,k} \leq t_s(O^{pos}) \quad \forall \ as(pos) = a \tag{6.1}$$

Da alle Positionen eines Artikels in der Kommissionier-Stufe nacheinander (d.h. ohne Rücklagerung der Fördereinheit) abgearbeitet werden, können die Operationen zu

$$O^a_{kom} = \mathcal{P}(O^{pos} \mid as(pos) = a) \tag{6.2}$$

zusammengefaßt werden. Für die Dauer gilt

$$pt(O^a_{kom}) = \sum_{pos \mid as(pos)=a} pt(O^{pos}) \tag{6.3}$$

Constraint (6.1) läßt sich somit umschreiben zu:

$$t_e(O^a_{hrl}) + t_{g,k} \leq t_s(O^a_{kom}) \tag{6.4}$$

Ist gewährleistet, daß die nachgefragten Artikel rechtzeitig an den Kommissionierplätzen bereitgestellt werden können, so können die Auslagerungs-Operationen O^a_{hrl} und damit Constraint (6.4) vernachlässigt werden. In Kapitel 6.3.2 gilt diese Vereinfachung nicht, da hier die Anzahl der zu befriedigenden Positionen eine Entscheidung darstellt.

Die zu verpackenden Artikel werden gepuffert, bis alle Positionen für ein Packstück vorhanden sind. Ab diesem Zeitpunkt plus der Transportzeit zu den Packplätzen kann mit dem Verpacken begonnen werden. Wird beispielsweise ein automatisches Behälterlager (ABL) für die Pufferung eingesetzt, kann es sinnvoll sein, die letzte Position eines Auftrages nicht in das ABL ein- und sofort wieder auszulagern, sondern auf direktem Weg zu den Packplätzen zu transportieren. Stellt $t_{k,p_{kp}}$ die Zeit dar, die für diesen Transportvorgang benötigt wird und O^j die Operation zum Verpacken des Auftrages, so muß gelten

$$t_e(O^a_{kom}) + t_{k,p_{kp}} \leq t_s(O^j) \quad \forall\, j \in \mathcal{J} \mid aj(j) = a \tag{6.5}$$

Diese Modellierung ist ausreichend, wenn davon ausgegangen wird, daß der Puffer zwischen Kommissionierung und Packerei zu keinem Zeitpunkt einen Engpaß darstellt. Wird der Puffer für eine realitätsnahe Betrachtung integriert, teilt sich die Fahrzeit $t_{k,p}$ in die Fahrzeit von den Kommissionierplätzen zum Puffer $t_{k,b_{kp}}$ und vom Puffer zu den Packplätzen $t_{b_{kp},p}$ auf. Die Kapazitätsbedingung für puffernde Ressourcen, Gleichung (4.17) muß erfüllt sein. Übertragen auf die beschriebene Anwendung ergibt sich mit der Puffer-Operation $O^a_{b_{kp}}$, die hier auf Artikelebene

definiert ist, und der Kapazität des Puffers $cp(b_{kp})$ das folgende Constraint:

$$t_e(O^a_{kom}) + t_{k,b_{kp}} \leq t_s(O^a_{b_{kp}}) \tag{6.6}$$

$$t_e(O^a_{b_{kp}}) + t_{b_{kp},p} \leq t_s(O^j) \tag{6.7}$$

$$\sum_{O^a_{b_{kp}}|\alpha} sz(O^a_{b_{kp}}) \leq cp(b_{kp}) \quad \forall\, 0 \leq t \leq H \tag{6.8}$$

$$\alpha \Leftrightarrow O^a_{kom} \succ O^a_{b_{kp}} \succ O^j |\ aj(j) = a \wedge t_e(O^a_{kom}) + t_{k,b_{kp}} < t \wedge t < t_s(O^j) - t_{b_{kp},p}$$

Die folgende Stufe, das Zusammenstellen der Touren, läßt sich ähnlich modellieren. Auch diese beiden Stufen sind wiederum durch einen Puffer b_{pv} entkoppelt. Die Transportzeit von der Packerei zu dem Puffer sei $t_{p,b_{pv}}$, von dem Puffer zum Versand $t_{b_{pv},v}$. Mit der Operation O^t für das Zusammenstellen der Touren ergibt sich

$$t_e(O^j) + t_{p,b_{pv}} \leq t_s(O^j_{b_{pv}}) \tag{6.9}$$

$$t_e(O^j_{b_{pv}}) + t_{b_{pv},v} \leq t_s(O^t) \tag{6.10}$$

$$\sum_{O^j_{b_{kp}}|\alpha} sz(O^j_{b_{pv}}) \leq cp(b_{pv}) \quad \forall\, 0 \leq t \leq H \tag{6.11}$$

$$\alpha \Leftrightarrow O^j \succ O^j_{b_{kp}} \succ O^t \wedge at(t) = j \wedge t_e(O^j) + t_{p,b_{pv}} < t \wedge t < t_s(O^t) - t_{b_{pv},v}$$

Für die Touren sind Fertigstellungszeitpunkte und Zeitpunkte des Auftragseingangs gegeben. Bei der folgenden Untersuchung wird vorausgesetzt, daß alle Aufträge zum Zeitpunkt der Planung vorliegen. Für die Touren muß gelten:

$$t_e(O^t) \leq t^{due}(t) \tag{6.12}$$

Zur Bearbeitung der Operationen an den einzelnen Ressourcen wird zusätzlich Personal gebunden, der Bedarf für die Ausführung einer Operation ist jeweils eins. Die angebotene Kapazität des Personal-Pool ist dynamisch, die der übrigen Ressourcen wird als konstant angenommen. Der Personal-Pool muß modelliert werden, wenn gilt:

$$cp(h,t) \leq \sum_{r \in r_p} cp(r) \quad \forall\, 0 \leq t \leq H \tag{6.13}$$

Die folgenden Constraints fassen die Ressourcen und den Personal-Pool zusammen, mit Constraint (4.14) gilt:

$$\sum_{O^a_{kom}|ra(O^a_{kom})=k \wedge t_s(O^a_{kom}) \leq t \leq t_e(O^a_{kom})} sz(O^a_{kom}, k) \leq cp(k) \; \forall \; 0 \leq t \leq H \tag{6.14}$$

$$\sum_{O^j|ra(O^j)=p \wedge t_s(O^j) \leq t \leq t_e(O^j)} sz(O^j, p) \leq cp(p) \; \forall \; 0 \leq t \leq H \tag{6.15}$$

$$\sum_{O^t|ra(O^t)=v \wedge t_s(O^t) \leq t \leq t_e(O^t)} sz(O^t, v) \leq cp(v) \; \forall \; 0 \leq t \leq H \tag{6.16}$$

$$\sum_{o|ra(o)=rs_p \wedge t_s(o) \leq t \leq t_e(o)} sz(o, rs_p) \leq cp(h, t) \; \forall \; 0 \leq t \leq H \tag{6.17}$$

6.3.2. Fall 2 - Keine Beachtung der Batchzuweisung

Wird in einem mehrstufigen Kommissioniersystem nach der in Kapitel 6.3.1 beschrieben Strategie verfahren, kann es zu Überlastungen in den zwischengelagerten Puffern kommen (siehe auch Abbildung 6.2). Die Konsequenz: Fertigstellungszeitpunkte lassen sich nicht einhalten. Die artikelorientierte Kommissionierung im ersten Schritt ist flexibler, wenn nur ein Teil der Positionen eines Artikels in einem Schritt kommisioniert werden. Somit werden Aufträge, die zu einem späteren Zeitpunkt versandfertig sein müssen, auch später fertiggestellt. Allerdings sind hier die komplexen Abhängigkeiten zwischen der Auftragsstruktur und den begrenzten Kapazitäten zu berücksichtigen.

Es ist zusätzlich die Frage zu beantworten, welche Positionen des gleichen Artikels zusammen kommissioniert werden sollten.

Zwischen der Bearbeitung zweier Positionen des gleichen Artikels sind Übergangszeiten $t^{set}(a, a)$ zu berücksichtigen. Diese können beispielsweise für das Bereitstellen eines neuen Behälters und die Quittierung der letzten fertiggestellten Position nötig sein. Die Zeiten sind unabhängig von den Artikeln und werden daher vereinfachend als t^{set}_g für den Übergang zwischen $g$leichen Artikeln bezeichnet. Zwischen der Kommissionierung zweier unterschiedlicher Artikel ist eine größere Übergangszeit $t^{set}(a, b) > t^{set}(a, a)$ vorzusehen, da hier in der Regel die bearbeitete Fördereinheit ausgeschleust und eine neue bereitgestellt werden muß. Diese Zeit ist ebenfalls von den Artikeln unabhängig und wird daher als Übergangszeit für $v$erschiedene Artikel als t^{set}_v bezeichnet. Wird eine Palette wieder in das Regal rückgelagert, kann erst wieder nach einer gegebenen (minimalen) Transportzeit $t_{k,g,k}$ (KOM $\rightarrow$ Regal $\rightarrow$ KOM) mit der Kommissionierung der Artikel fortgefahren werden. Sei n die Anzahl der Positionen, die Artikel a enthalten

$n = |P(\mathcal{POS})| \ \forall \ as(pos) = a$, so sind $1, \ldots, n$ Transportoperationen nötig, abhängig davon, wie viele Positionen des Artikels in der Stufe zusammen kommissioniert werden. Bei dem in Kapitel 6.3.1 beschriebenen Fall ist ein einziger Transport, bei einer auftragsorientierten Kommmissionierung sind n Transporte nötig. Diese reihenfolgeabhängige Anzahl von Operationen kann mit Hilfe der Übergangszeiten modelliert werden.

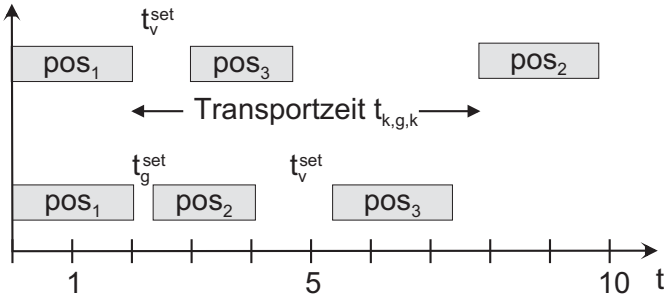


Abbildung 6.8.: Übergangszeiten und Transportzeit beim Kommissionieren.

In Abbildung 6.8 sind unterschiedliche Reihenfolgen und damit Übergangszeiten für zwei Artikel a, b, die in drei Positionen pos_1, pos_2, pos_3 kommissioniert werden, dargestellt. Position pos_1 und pos_2 sind Artikel a, pos_3 Artikel b zugeordnet. Die bearbeitenden Operationen sind O^{pos_1} für pos_1, etc. Folgende Reihenfolgen können unterschieden werden:

- $pos_1 \succ pos_2 \succ pos_3 \Rightarrow a \succ a \succ b$. Die Übergangszeiten zwischen den gleichen und den unterschiedlichen Artikeln müssen eingehalten werden (in Abbildung 6.8 unten dargestellt), es ergibt sich

$$t_e(O^{pos_1}) + t_g^{set} \leq t_s(O^{pos_2}) \wedge t_e(O^{pos_2}) + t_v^{set} \leq t_s(O^{pos_3}) \tag{6.18}$$

- $pos_1 \succ pos_3 \succ pos_2 \Rightarrow a \succ b \succ a$. Die Übergangszeiten zwischen den unterschiedlichen Artikeln müssen eingehalten werden, die Restmenge des Artikels a wird nach der Bearbeitung wieder zurückgelagert, somit ist eine zusätzliche Transportzeit $t_{k,g,k}$ zu berücksichtigen (in Abbildung 6.8 oben dargestellt), es ergibt sich

$$t_e(O^{pos_1}) + t_v^{set} \leq t_s(O^{pos_3}) \wedge t_e(O^{pos_3}) + t_v^{set} + t_{k,g,k} \leq t_s(O^{pos_2}) \tag{6.19}$$

  Diese Reihenfolge wäre nur sinnvoll, wenn $t_{k,g,k} < pt(O^{pos_3})$, andernfalls kommt es zu Stillstandszeiten am Kommissionierplatz.

Wie leicht zu sehen ist, können die beiden Positionen pos_1 und pos_2 ausgetauscht werden, die Gleichungen behalten ihre Gültigkeit.

Die Übergangszeiten für gleiche und unterschiedliche Artikel werden um die Transportzeit erweitert. Diese ist zu berücksichtigen, wenn an einem Kommissionierplatz zwischen zwei beliebigen Artikeln ein oder mehrere andere Artikel kommissioniert werden. Formal ausgedrückt muß gelten:

$$\begin{aligned} & t^{set}(O^{pos_1}, O^{pos_n}) = t_{k,g,k} \Leftrightarrow t_e(O^{pos_1}) + t_{k,g,k} \leq t_s(O^{pos_n}) \\ & \forall \quad O^{pos_1} \succ O^{pos_2} \ldots \succ O^{pos_{n-1}} \succ O^{pos_n} \wedge \\ & as(pos_1) = a \wedge as(pos_2) \neq a \wedge \ldots \wedge as(pos_{n-1}) \neq a \wedge as(pos_n) = a \end{aligned} \tag{6.20}$$

Hier wird angenommen, daß bei parallelen Kommissionierplätzen nicht der Fall auftritt, daß der Artikel an einem Platz kommissioniert und nach Fertigstellung auf dem nächsten Kommissionierplatz wieder zum Kommissionieren ausgeschleust wird.

6.4. Lösen des COP

Die Entscheidungsvariablen des Optimierungsproblemes sind die Startzeitpunkte t_s der Operationen der einzelnen Stufen. Der Wertebereich ist initial gegeben zu $D_{t_s} = [0, \ldots, H]$ wird aber durch die definierten Constraints, d.h. den Auftragseingangszeitpunkt und den gewünschten Abfahrzeitpunkt eingeschränkt.

6.4.1. Zielfunktion

Das Ziel bei der Bestimmung einer optimalen Reihenfolge ist die Einhaltung der vorgegebenen Versandzeiten zur Maximierung des Servicegrades. Das nicht relaxierte Constraint aus Gleichung (6.12) läßt keine Verspätung zu. Diese Bedingung ist in praxi zu hart. Eine minimale unvermeidbare Verspätung würde zu einer noch tolerierbaren verspäteten Auslieferung führen. Daher wird das Constraint relaxiert zu

$$t_e(O^t) \leq t^{due}(t) + l^t \; \forall \; t \tag{6.21}$$

Der Term l^t wird als die Verspätung (engl. lateness) von Tour t bezeichnet, der Wertebereich ist $D_{l^t} = \{-\infty, \infty\}$. Das definierte CSP ist somit in jedem Fall lösbar.

Eine Zielfunktion, in der die (gewichtete) Summe der Verspätungen minimiert wird, priorisiert die wichtigen Touren, was jedoch zu sehr großen Verspätungen

bei den unwichtigen Touren führen kann. Um diese Dominanzeffekte zu vermeiden wurde hier die Minimierung der *maximalen* Verspätung $L_{max} = max(l^t) \quad \forall\, t$ aller Touren als Zielkriterium gewählt, $Z = \text{MIN } L_{max}$.
Das COP wird durch Hinzufügen der folgenden Constraints formuliert:

$$l^t \leq L_{max} \quad \forall\, t \tag{6.22}$$

Für die praktische Anwendung weniger relevant, aus der Literatur aber hinreichend bekannt, ist das Ziel, die Gesamtdurchlaufzeit C_{max} des Schedules zu minimieren. Integriert wird die Zielfunktion $Z = \text{MIN } C_{max}$ in das COP durch das Constraint

$$t_e(O^t) \leq C_{max} \quad \forall\, t \tag{6.23}$$

Die vorgegebenen Fertigstellungszeitpunkte wurden hier ebenfalls relaxiert und treten somit in den Hintergrund. Liegen die Aufträge früh genug vor (d.h. am Tag vor dem gewünschten Fertigstellungszeitpunkt), ist also die Erfüllung nicht kritisch, kann durch die Verwendung dieser Zielfunktion die Auslastung der Ressourcen maximiert werden.

6.4.2. Beeinflussung der Suche

Explizite Orderings

Durch die Definition von expliziten Orderings kann der Suchraum signifikant verkleinert werden. Werden die Artikel im Fall 1 nacheinander kommissioniert, existieren bei einem Artikel a, der in n Positionen kommissioniert wird, $n!$ unterschiedliche Permutationen. Die Gesamtdauer $pt(O^a_{kom})$, ist jedoch immer gleich. Wird als Ziel MIN L_{max} verfolgt, so sollten die Positionen nach aufsteigendem $t_e(O^{pos})$ bearbeitet werden. Die Verwendung von t_e setzt selbstverständlich die Einschränkung der Wertebereiche durch eine Konsistenzprüfung und Sicherstellung voraus.
Ähnliche Überlegungen lassen sich für Fall 2 anstellen. Hier sind die Positionen nicht explizit über die Artikel aneinander gebunden, jedoch sollte auch hier bei einem Artikel, der in mehreren Positionen kommissioniert wird, diejeninge mit dem kleinsten $t_e(O^{pos})$ zuerst bearbeitet werden.
Die folgenden Constraints können zusätzlich definiert werden

$$O^{pos_2} \succ O^{pos_1} \quad \forall\, as(pos_1) = as(pos_2) \wedge t^{due}(O^{pos_2}) > t^{due}(O^{pos_1}) \tag{6.24}$$

Bei der folgenden Stufe, der Packerei, gilt ähnliches. Allerdings ist hier zu beachten, daß durch die Verwendung von Ressourcen mit $cp(r) > 1$ mehrere Packstücke zugleich bearbeitet werden können. Somit wird die Reihenfolge von zwei Packstücken ps_1 und ps_2 nicht über die Verknüpfung von Start- und Endzeitpunkt, sondern über die beiden Startzeitpunkte realisiert:

$$t_s(O^{ps_1}) \geq t_s(O^{ps_2}) \quad \forall\, t^{due}(O^{pos_2}) > t^{due}(O^{pos_1}) \tag{6.25}$$

Es hat sich herausgestellt, daß diese expliziten Orderings nicht immer zu Laufzeitverbesserungen führen. Bei dem Suchprozeß steigt durch jedes zusätzliche Constraint der Aufwand für die Sicherstellung der Konsistenz. Dies ist auch durch den Suchprozeß begründet, da bei Auswahl einer Variablen kanten- bzw. k-Konsistenz erreicht werden soll. Zusätzliche Constraints binden zusätzliche Variablen. War vorher die (einfache) Sicherstellung der Kantenkonsistenz ausreichend, so muß nun k-Konsistenz gewährleistet werden.

Redundante Constraints

Redundante Constraints werden hier als das explizite Definieren von impliziten Verknüpfungen verstanden. Aus dem Constraint $O^{pos} \succ O^{ps} \succ O^{t}$ kann beispielsweise die implizit angegebene Verknüpfung $O^{pos} \succ O^{t}$ explizit gemacht werden. Bei der Sicherstellung der Konsistenz kann dies zu Laufzeitverbesserungen führen.

6.5. Anwendung

Das Constraint Optimization Problem wurde mit den Klassenbibliotheken ILOG Solver 4.31 und Scheduler 4.3 in C++ umgesetzt. Die im folgenden beschriebenen Versuche wurden auf einem Pentium Pro Rechner mit 450 Mhz Taktfrequenz und 128 MB Hauptspeicher ausgeführt. Den Untersuchungen liegen drei Nachfrageszenarien unterschiedlicher Größe zugrunde. Das Verfahren zur Sortierung der Variablen wurde in Kapitel 5.4 beschrieben. Die Variable, die als erste im Schedule eingeplant werden kann und den kleinsten Fertigstellungszeitpunkt hat, wird ausgewählt. Der zugewiesene Wert ist der frühest mögliche Startzeitpunkt.

Zunächst wurde das Beispiel aus Abbildung 6.3 für die beiden Fälle untersucht. In Abbildung 6.9 ist der Bestandsverlauf der Puffer zwischen der Kommissionierstufe und der Packerei, sowie zwischen der Packerei und dem Versand dargestellt. Bei Fall 1 (Beachtung des Batches) werden die Artikel zusammengefaßt, nach 5 [ZE] wird eine Position in den Puffer eingelagert. Nachdem drei Positionen eines

jeden Auftrages im Puffer sind, kann die vierte sofort zu den Packplätzen durchgeschleust werden, daher ist der maximale Bestand des Puffers 12 (4 Aufträge à 3 Positionen). Während die Positionen der ersten beiden Aufträge verpackt werden, wird an den Kommissionierplätzen weitergearbeitet, was an dem wieder zunehmenden Bestand zu erkennen ist. Die geforderten Abfahrtszeitpunkte können nicht erfüllt werden, die Werte der Zielfunktion sind in Tabelle 6.1 dargestellt.

	Fall 1		Fall 2	
	C_{max}	L_{max}	C_{max}	L_{max}
$cp(k) = 1$	114	30	130	0
$cp(k) = 2$	64	-20	63	-30

Tabelle 6.1.: Zielfunktionswerte des Beispielproblems.

Wird bei der Optimierung die Zuweisung zu einem Batch nicht beachtet (Fall 2), ergibt sich der ebenfalls in Abbildung 6.3 dargestellte Bestandsverlauf. Hier werden die Positionen der Artikel der ersten Tour zusammengefaßt, nach dem Bearbeiten von zwei Positionen ist die Übergangszeit zu erkennen. Auch hier wird die vierte Position direkt zu den Packplätzen weitertransportiert. Der mittlere Bestand ist niedriger, die gewünschten Abfahrtszeiten können erfüllt werden, allerdings ist, aufgrund der größeren Anzahl an Übergängen, die Gesamtdurchlaufzeit C_{max} höher.

Sind auf der ersten Stufe zwei Ressourcen vorhanden, ergibt sich der in Abbildung 6.10 dargestellte Verlauf. Auch hier sind wieder die typischen Verläufe zu erkennen. Bei Fall 1 werden die Artikel zusammengefaßt, bei Fall 2 kann jede Position einzeln betrachtet werden.

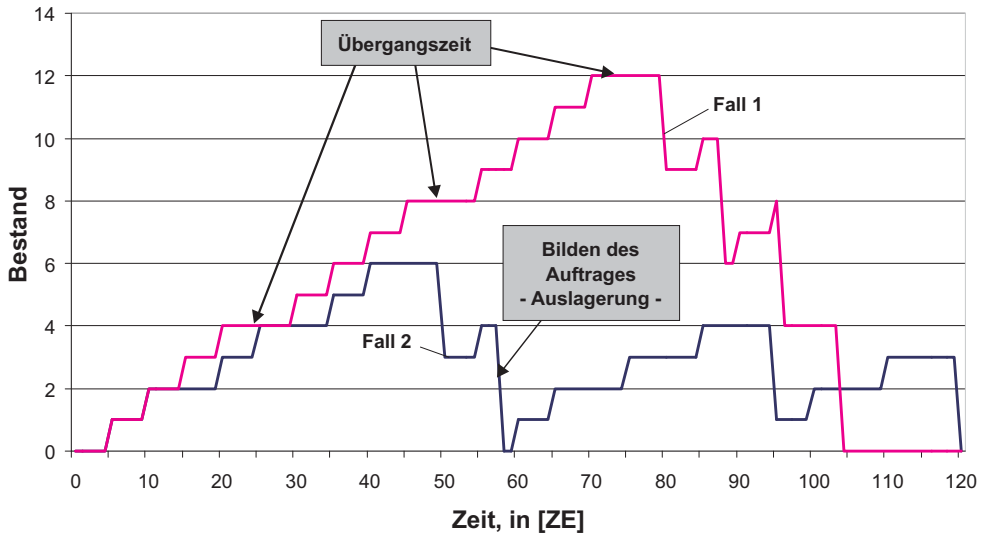


Abbildung 6.9.: Bestandsverlauf des Puffers b_{kp} zwischen Kommissionierbereich und Packerei für das Beispiel aus Abbildung 6.2, $Z = \text{MIN } L_{max}$, Fall 1 (Beachtung Batch) und Fall 2 (Nichtbeachtung Batch).

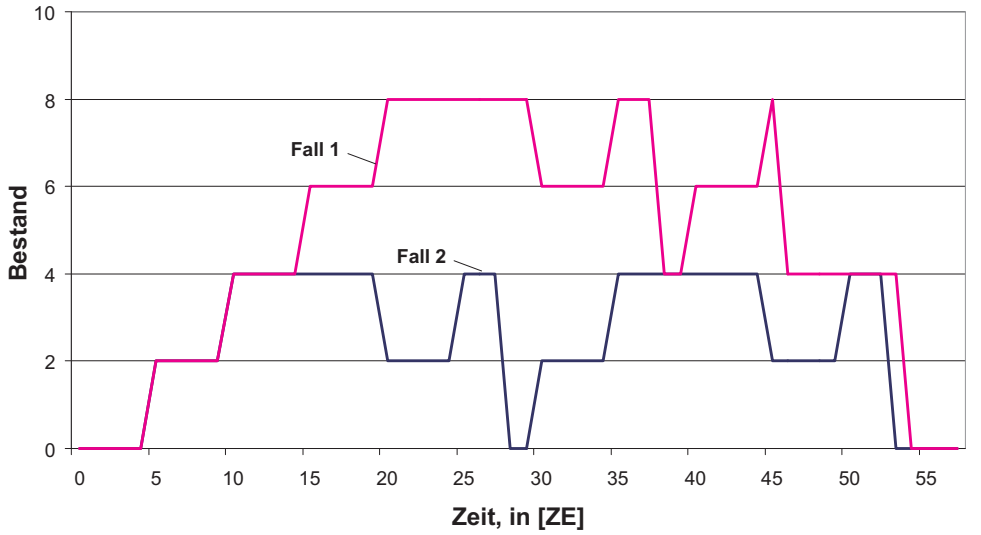


Abbildung 6.10.: Gleiche Daten wie in Abbildung 6.9, aber mit zwei Ressourcen auf der ersten (Kommissionier-)Stufe.

6.5.1. Szenarien

Drei unterschiedliche Szenarien dienen als Grundlage für die Experimente. Ein Szenario wird definiert durch die

- **Gesamtanzahl der Artikel**: In realen Distributionszentren sind teils mehr als 100.000 Artikel vorhanden. Hier werden nur die in dem Untersuchungszeitraum nachgefragten Artikel modelliert.

- **Anzahl der Touren**, die das Distributionszentrum in dem Untersuchungszeitraum verlassen.

- **maximale Anzahl der Aufträge pro Tour**, **maximale Anzahl der Positionen pro Auftrag**: Diese Parameter bestimmen die Struktur des Optimierungs-Problems und werden als gleichverteilte Zufallszahlen ermittelt.

- **maximal zu kommissionierende Menge (einer Position)**: Diese Menge bestimmt die Bearbeitungszeit der Artikel an den Kommissionierplätzen, die Menge wird als gleichverteilte Zufallszahl ermittelt.

- **Kapazitäten** der Kommissionier-, Pack- und Versandstufe (Anzahl Arbeitsplätzen) und Größe des Personalpools (Anzahl Mitarbeiter).

- **Bearbeitungszeiten** an den Kommissionier-, Pack-, und Versandplätzen.

- **Verteilung der Schnell- und Langsamdreher**: Einige Artikel werden häufiger nachgefragt, beispielsweise im Ersatzteilgeschäft, was sich auf die Kombinationsmöglichkeiten bei Untersuchung von Fall 2 (keine Beachtung der Batch-Zuweisung) auswirkt. Die Artikel werden nach einer einer AC-Verteilung in schnelldrehende (A) und langsamdrehende (C) aufgeteilt. Diese Einteilung hat Auswirkungen auf die Erzeugung der Aufträge.

- **Übergangszeiten** an den Kommissionierplätzen, $t_g, t_v, t_{k,g,k}$.

Die Daten wurden zufällig erzeugt. Die Struktur des Problems, d.h. die Anzahl der Aufträge pro Tour, die Anzahl der Positionen pro Auftrag und die zu kommissionierende Menge wurden nach einer Gleichverteilung erzeugt. Die Parameter geben die jeweiligen Maximalwerte an. Der Algorithmus zur Erzeugung der Szenarien ist in Anhang B.1 detailliert beschrieben.

Szen.	Artikel	Touren	Auftr. pro Tour	Pos. pro Auftrag	Auftr., ges.	Pos., ges.	$cp(k)$	$cp(p)$	$cp(t)$
1	10	3	4	6	4	12	2	2	1
2	50	3	6	10	31	163	2	2	1
3	100	5	10	10	68	224	5	3	2

Tabelle 6.2.: Bei der Untersuchung verwendete Nachfrageszenarien.

Für jede Tour ist eine unterschiedliche Abfahrtszeit, d.h. $t^{due}(t)$ angegeben. In praktischen Distributionszentren sind diese Abfahrtszeitpunkte nicht gleichverteilt, sondern liegen häufig innerhalb von Clustern. Die Abfahrzeitpunkte der Szenarien wurden nachbearbeitet, um diesen Effekt zu erhalten.

Die Daten der Szenarien sind in Tabelle 6.2 dargestellt.

Für die Szenarien wurde die Sensitivität der Laufzeit hinsichtlich folgender Parameter untersucht:

- der Einfluß von expliziten Orderings,
- die Kapazität der Puffer $cp(b_{kp})$, $cp(b_{pv})$ und des Personalpools $cp(h)$,
- die Unterscheidung in Fall 1 oder Fall 2,
- die Optimierung nach der Zielfunktion $Z = \text{MIN } L_{max}$ oder nach $Z = \text{MIN } C_{max}$.

Die Untersuchung wurde im Gegensatz zu der noch folgenden Anwendung in Kapitel 7 nicht mit Hilfe eines Experimentendesigns durchgeführt, da nicht die *Konfiguration* des Systems, sondern die Laufzeiten des Verfahrens untersucht werden sollten. Der Wert der Zielfunktion wird als Bewertung verwendet, kann aber nicht zum Vergleich der unterschiedlichen Szenarien herangezogen werden, da die Strukturen der zugrunde liegenden Probleme unterschiedlich sind.

Beispielhafte Darstellung von Szenario 3

Die Ergebnisse werden im folgenden anhand von Szenario 3 dargestellt und erläutert. Szenario 1 und Szenario 2 weisen vergleichbare Verläufe auf, allerdings sind die Effekte bei dem größten Szenario 3 besser zu erkennen. In Abbildung 6.11 ist der Verlauf des Pufferbestandes für Szenario 3 dargestellt, die Batch-Zuweisung wird nicht beachtet, die Kapazität der Ressourcen ist in Tabelle 6.2 angegeben. Deutlich ist zu erkennen, wie die kommissionierten Positionen in den

Puffer eingelagert werden. Ist ein Auftrag komplett, werden die Positionen ausgelagert, der Auftrag komplettiert und in den Puffer zwischen Packerei und Versand eingelagert, beispielsweise zum Zeitpunkt $t = 52$ oder $t = 70$. Die Touren werden entsprechend ihrer gewünschten Fertigstellungszeitpunkte komplettiert, der maximale Pufferbestand ist 63.

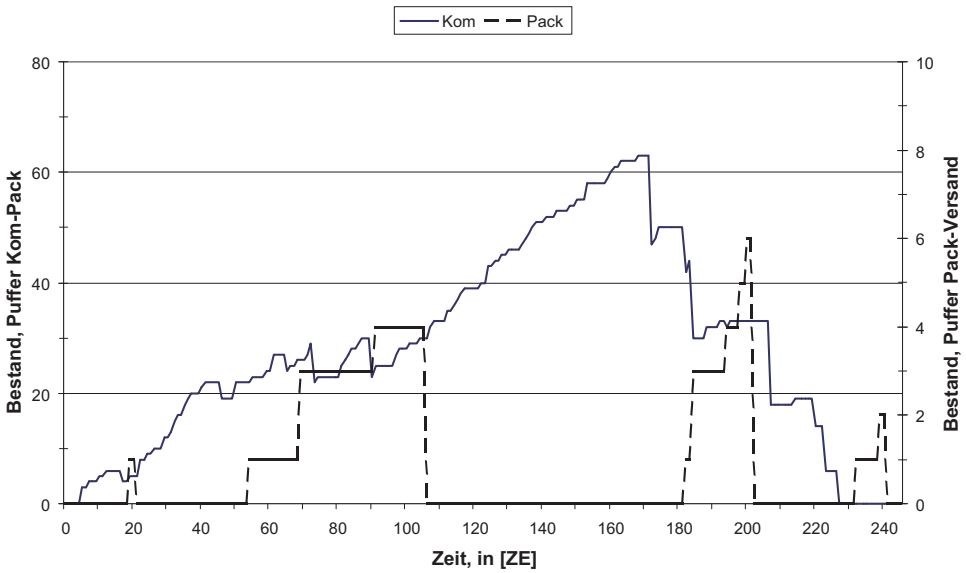


Abbildung 6.11.: Verlauf des Pufferbestandes, zugrunde liegt Szenario 3, keine Begrenzung des Puffers und des Personalpools, keine Beachtung der Batch-Zuweisung.

Abbildung 6.13 zeigt die Belegung des Puffers, wenn die Batch-Zuweisung beachtet wird. Die Grafiken 6.13 und 6.11 sind sehr ähnlich, was auf die Verteilung der Artikel auf die Positionen zurückzuführen ist, die in Abbildung 6.12 dargestellt ist.

In Abbildung 6.14 wurde $Z = \text{MIN } C_{max}$ als Zielfunktion verwendet. Deutlich ist zu erkennen, daß die Übergangszeiten vermieden werden, die Fertigstellung der Aufträge erfolgt zu einem späteren Zeitpunkt. Der Puffer wird stärker belastet, der maximale Bestand liegt bei 91.

Die Nutzung des Personal-Pools zeigt einen ähnlichen Verlauf wie der Bestand der Puffer. Wird als Zielfunktion MIN L_{max} verwendet, ergeben sich für Beachtung und Nichtbeachtung der Batch-Zuweisung sehr ähnliche Verläufe (Abbildung 6.15 und 6.16). Bei Verwendung der Zielfunktion MIN C_{max} wird das Personal anfangs nur bei den Kommissionierplätzen, später verstärkt bei den Packplätzen und im Versandbereich eingesetzt (Abbildung 6.17).

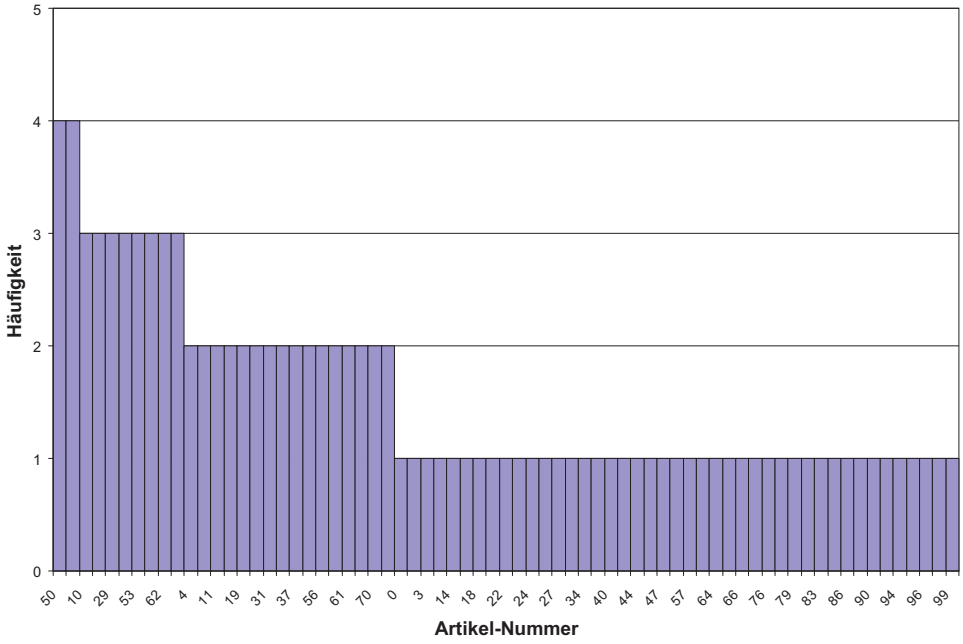


Abbildung 6.12.: Zugriffshäufigkeit der Artikel für Szenario 3.

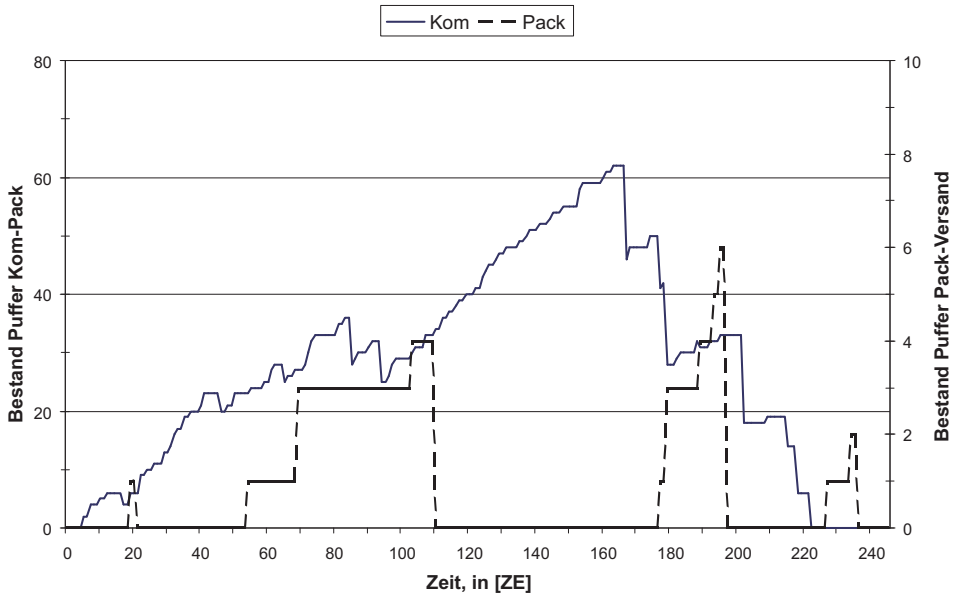


Abbildung 6.13.: Verlauf des Pufferbestandes, zugrunde liegt Szenario 3, keine Begrenzung des Puffers und des Personalpools, Beachtung der Batch-Zuweisung.

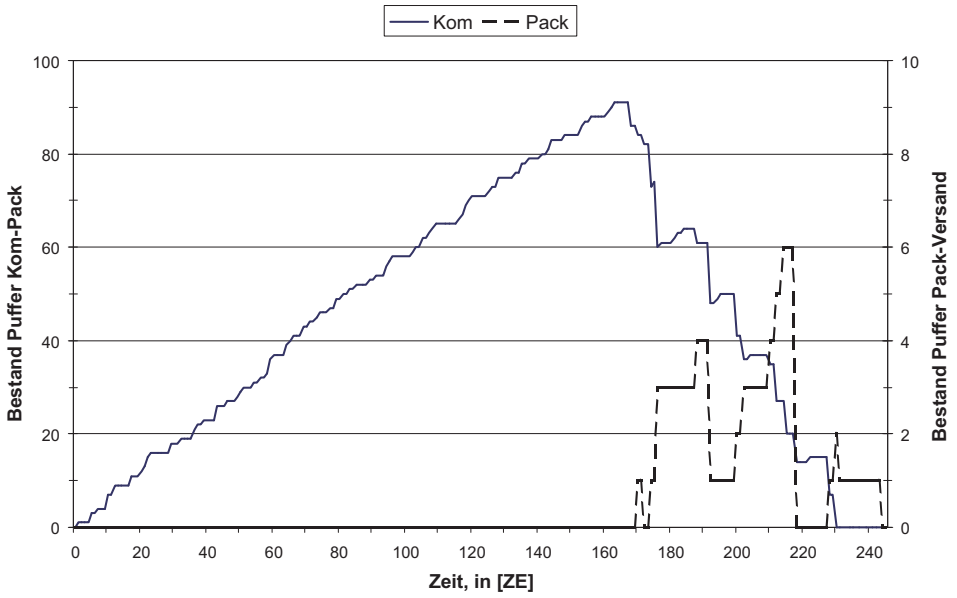


Abbildung 6.14.: Verlauf des Pufferbestandes, zugrunde liegt Szenario 3, keine Begrenzung des Puffers und des Personalpools, keine Beachtung der Batch-Zuweisung, die Zielfunktion ist $Z = \text{Min } C_{max}$.

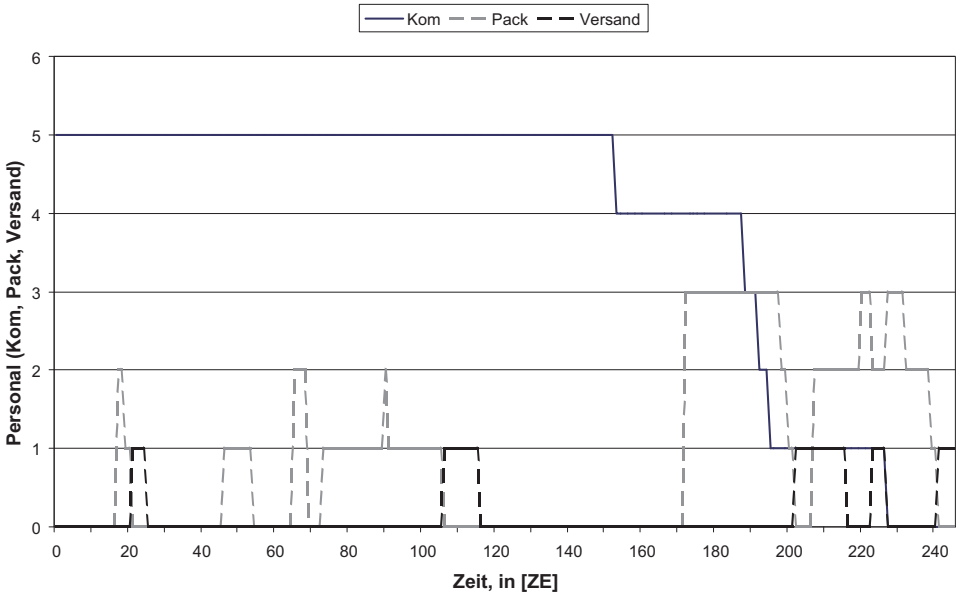


Abbildung 6.15.: Nutzung des Personal-Pools, zugrunde liegt Szenario 3, keine Begrenzung des Personalpools, keine Beachtung der Batch-Zuweisung.

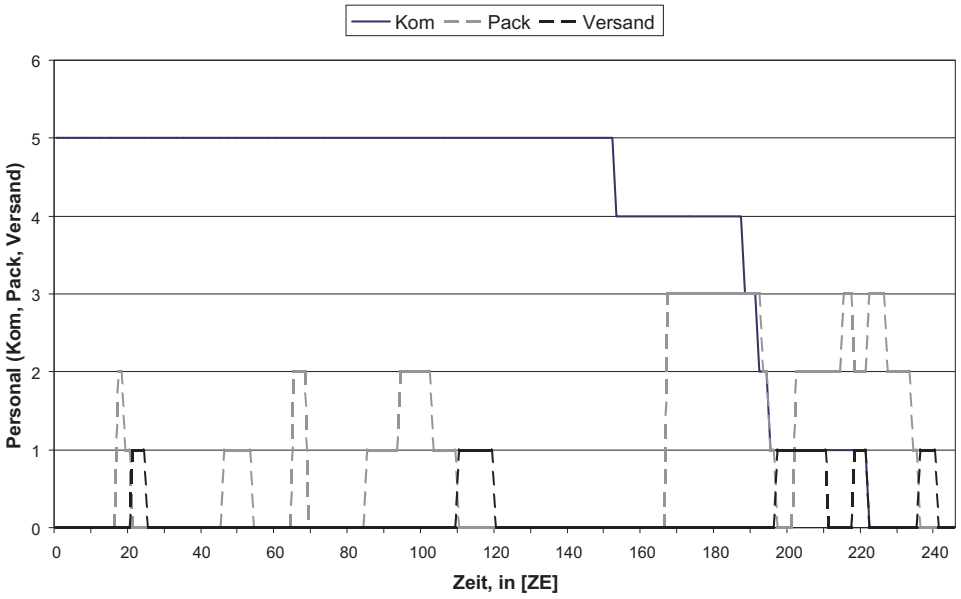


Abbildung 6.16.: Nutzung des Personal-Pools, zugrunde liegt Szenario 3, keine Begrenzung des Personalpools, Beachtung der Batch-Zuweisung.

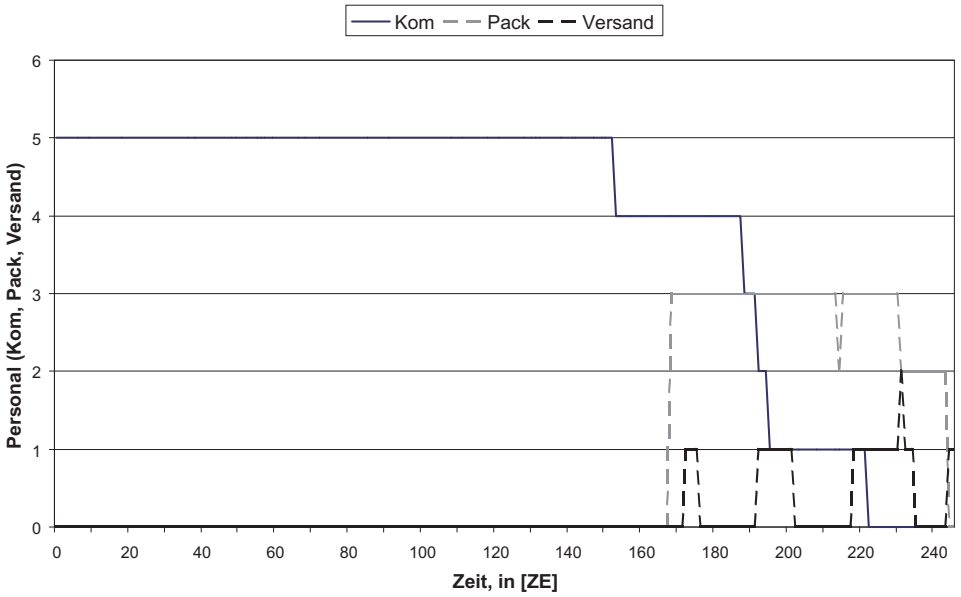


Abbildung 6.17.: Nutzung des Personal-Pools, zugrunde liegt Szenario 3, keine Begrenzung des Personalpools, keine Beachtung der Batch-Zuweisung, Ziel MIN C_{max}.

Explizite Orderings

Der Einfluß von expliziten Orderings wurde mit zwei zufälligen Szenarien mit jeweils fünf unterschiedlichen Konfigurationen der Fertigstellungszeitpunkte durchgeführt. Die Probleme wurden mit dem in Anhang B.1 beschriebenen Algorithmus erzeugt. Die unterschiedlichen Konfigurationen bei gleicher Struktur des Problems wurden durch Variation des Bereiches der gewünschten Fertigstellungszeitpunkte erreicht. Das kleine Problem bestand aus insgesamt 16 Positionen, und 3 Touren. Das große Problem bestand aus insgesamt 113 Positionen, 33 Aufträgen und 9 Touren. Die Ergebnisse der Testläufe sind in Tabelle 6.3 zusammengefaßt. Die Problemstruktur bedingt, daß nur wenige zusätzliche redundante Constraints definiert werden können, daher wurde hier auf umfangreiche Tests verzichetet.

		Fall 1			Fall 2		
		$\bar{t}_{cpu}$	$\sigma(t_{cpu})$	# Lsg.	$\bar{t}_{cpu}$	$\sigma(t_{cpu})$	# Lsg.
kl. Problem	ohne expl. Ord.	3.0	0.4	6	2.66	1.06	6
	einfache expl. Ord.	4.0	8.8	6	2.66	0.66	6
	komplexe expl. Ord.	2.833	0.166	6	2.83	0.16	6
	alle expl. Ord.	2.5	0.7	6	2.166	0.166	6
gr. Problem	ohne expl. Ord.	235	61990	10 (*10*)	2.5	2.5	10 (*5*)
	einfache expl. Ord.	55	2250	10	30.4	960	10 (*5*)
	komplexe expl. Ord.	64.9	3110	10	36.5	1400	10 (*5*)
	alle expl. Ord.	61.2	2733	10	34.1	1217	10 (*1*)

Tabelle 6.3.: Beeinflussung der Lösungssuche durch Definition expliziter Orderings. Angegeben sind die Laufzeiten zur Bestimmung der Lösung. Bei der Anzahl Lösungen ist in Klammern angegeben, wie oft die Laufzeitbegrenzung erreicht wurde, d.h. also die Optimalität nicht gezeigt werden konnte. Die Werte von L_{max} waren jedoch bei allen Replikationen (bezogen auf die Verteilung der Fertigstellungszeitpunkte) gleich.

Mit $\bar{t}_{cpu}$ wird die mittlere Laufzeit in [sec], mit $\sigma(t_{cpu})$ die Standardabweichung bezeichnet. Es wurden pro Problemgröße sechs unterschiedliche Verteilungen der Fertigstellungszeitpunkte unterstellt. Deutlich ist die Abnahme der Laufzeit zu erkennen, die besten Ergebnisse werden erzielt, wenn die einfachen *und* die komplexen zusätzlichen Constraints verwendet werden. Dies gilt für Fall 1 und Fall 2, allerdings ist bei dem großen Problem für Fall 2 eine Ausnahme zu erkennen.

Puffergröße

Die Puffergröße läßt sich nicht als definierter (minimaler) Wert angeben, daher wurde der Einfluß der Puffergröße auf die Laufzeit detailliert untersucht. Zugrunde liegt Szenario 3, die Kapazitäten der Puffer wurden wie in Tabelle 6.4 dargestellt verändert. Die Kapazitätsgrenze wurde zu 80% (65%) der maximalen Belastung bei unendlicher Pufferkapazität festgelegt. Die Zielfunktion war $Z = \text{MIN } L_{max}$, die Ergebnisse sind in Tabelle 6.4 dargestellt.

			Fall 1		Fall 2	
Versuch	$cp(b_{kp})$	$cp(b_{pv})$	t_{cpu} [sec]	L_{max}	t_{cpu} [sec]	L_{max}
1	∞	∞	16.6	-104	23.4	-104
2	50 (80%)	∞	9478	-104	9615	-104
3	∞	5	17.0	-104	23.9	-104
4	50	5	9591	-104	9748	-104
5	40 (65 %)	∞	*724*	-92	21900	-104
6	∞	4	keine Lösung			
7	40	4	736	-92	13890	-104

Tabelle 6.4.: Versuche zur Untersuchung der Sensitivität des Puffers.

Die Begrenzung des Puffers führt zu sehr ähnlichen Bestandsverläufen für Beachtung und Nicht-Beachtung der Batch-Zuweisung, wie in Abbildung 6.18 und 6.19 zu erkennen ist. Die Zielfunktionswerte unterscheiden sich nur marginal, durch das Verfahren konnten die Artikel in beiden Fällen in eine (sub-)optimale Reihenfolge gebracht werden. Bedingt durch die relativ hohe Anzahl an Artikeln im Vergleich zu Positionen wirken sich hier die Effekte der Beachtung zu Nicht-Beachtung der Batch-Zuweisung nicht so stark aus.

Die Belastung des Personalpools ist in den Abbildungen 6.20, 6.21 aufgeteilt nach Stufen dargestellt. Die Verläufe für Fall 1 und Fall 2 sind sehr ähnlich. Interessant ist die ungleichmäßige Verteilung des Personals. Am Anfang werden die Kommissionierplätze komplett belegt, nachdem einige Positionen in die Puffer eingelagert wurden, ziehen die Packplätze Personal zum Komplettieren der Aufträge an. Schließlich werden die Aufträge zu Touren zusammengestellt. Die Ressourcen der Kommissionierstufe sind anfangs ($t = [0, \ldots, 110]$) der Engpaß des Systems, anschließend ist der Puffer die begrenzende Ressource, bis im Bereich $t = [168, \ldots, 182]$ die Ressourcen der Packstufe den Engpaß darstellen. Die Personalressource ist immer ausreichend vorhanden. Der schwankende Verlauf ab $t = 110$ ist auf den begrenzten Puffer zurückzuführen. Die Kommissionierstufe kann keine Positionen mehr einlagern, somit müssen zuerst die Aufträge komplet-

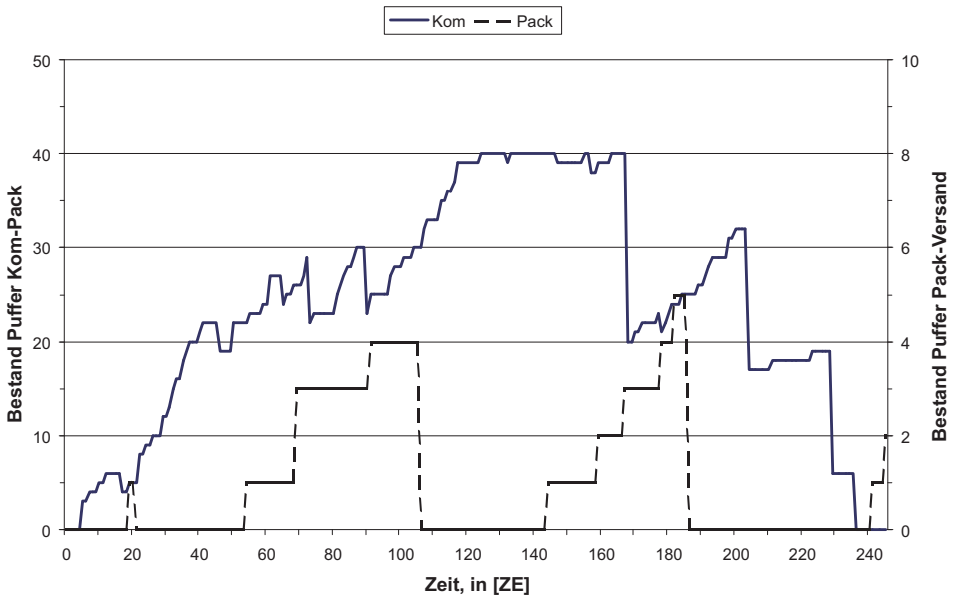


Abbildung 6.18.: Verlauf des Pufferbestandes bei Begrenzung des Puffers auf $cp(b_{kp}) = 40$, $cp(b_{pv}) = \infty$, keine Beachtung Batch-Zuweisung.

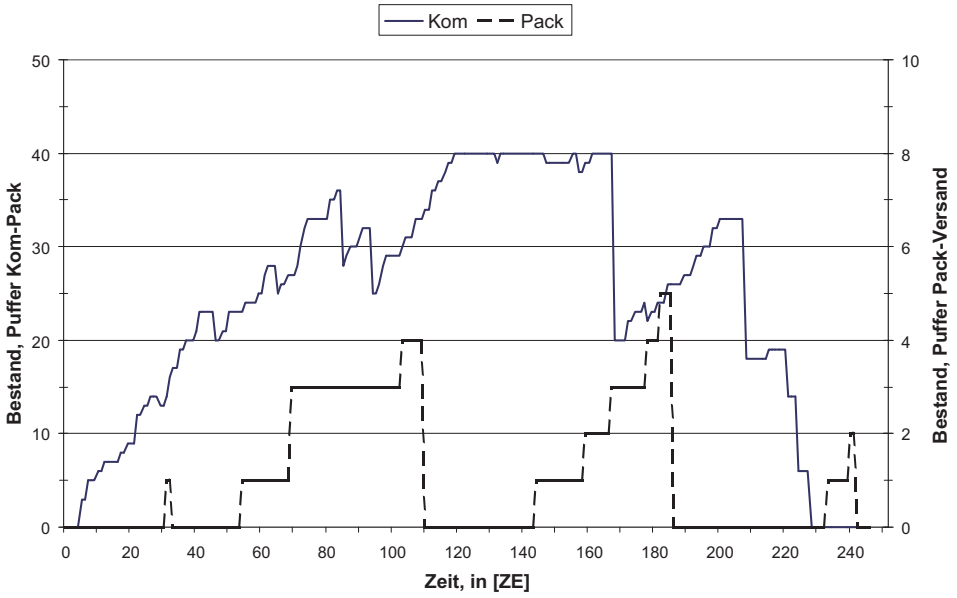


Abbildung 6.19.: Verlauf des Pufferbestandes bei Begrenzung des Puffers auf $cp(b_{kp}) = 40$, $cp(b_{pv}) = \infty$, Beachtung Batch-Zuweisung.

tiert werden, hier allerdings stehen nur $cp(p) = 3$ Plätze zur Verfügung. Hieraus resultiert der schwankende Verlauf der Personalnutzung der Kommissionier- und Packstufe.

In den Abbildungen 6.22 und 6.23 ist nochmals die gesamte Nutzung des Personalpools aufgezeigt, auch hier ist der Einbruch aufgrund des begrenzten Puffers deutlich zu erkennen, die starken Schwankungen der einzelnen Stufen gleichen sich jedoch in der summarischen Betrachtung aus.

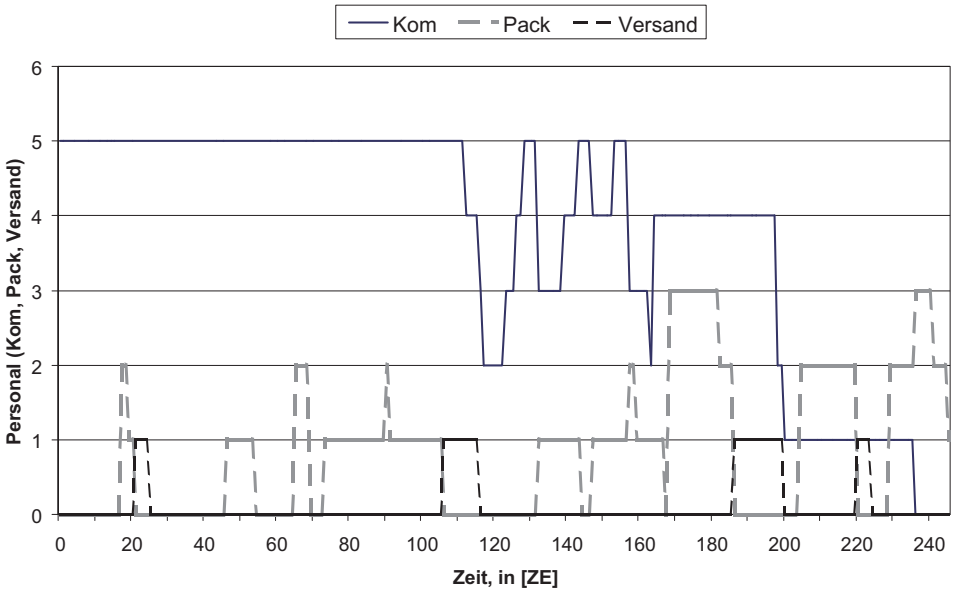


Abbildung 6.20.: Verteilung des Personaleinsatzes bei Begrenzung des Puffers auf $cp(b_{kp}) = 40$, $cp(b_{pv}) = \infty$, nach Stufen, keine Beachtung Batch-Zuweisung.

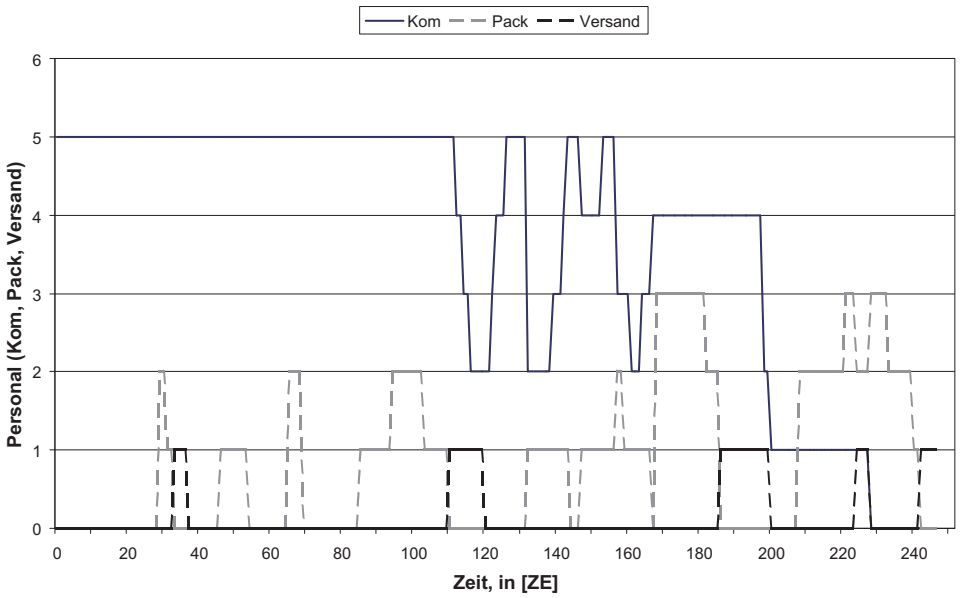


Abbildung 6.21.: Verteilung des Personaleinsatzes bei Begrenzung des Puffers auf $cp(b_{kp}) = 40$, $cp(b_{pv}) = \infty$, nach Stufen, Beachtung Batch-Zuweisung.

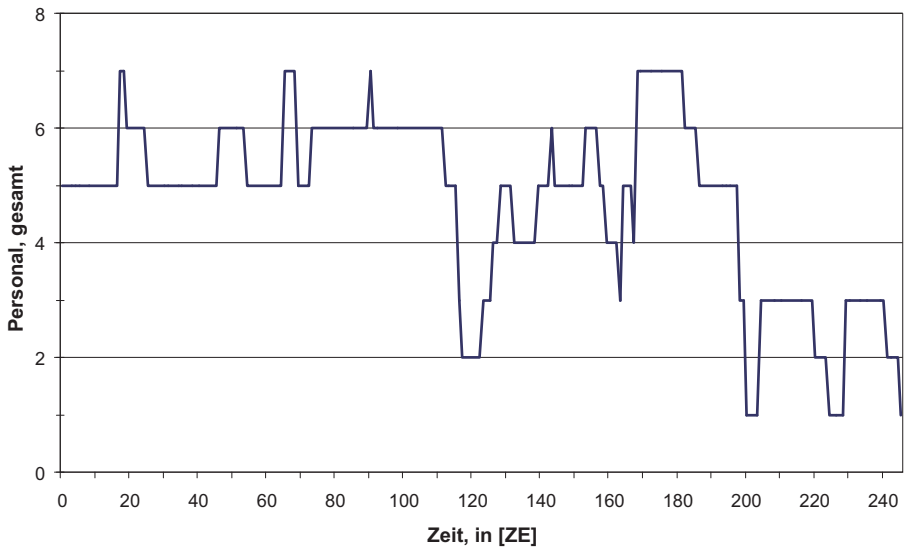


Abbildung 6.22.: Verteilung des Personaleinsatzes bei Begrenzung des Puffers auf $cp(b_{kp}) = 40$, $cp(b_{pv}) = \infty$, keine Beachtung Batch-Zuweisung. Dargestellt ist der Personalpool.

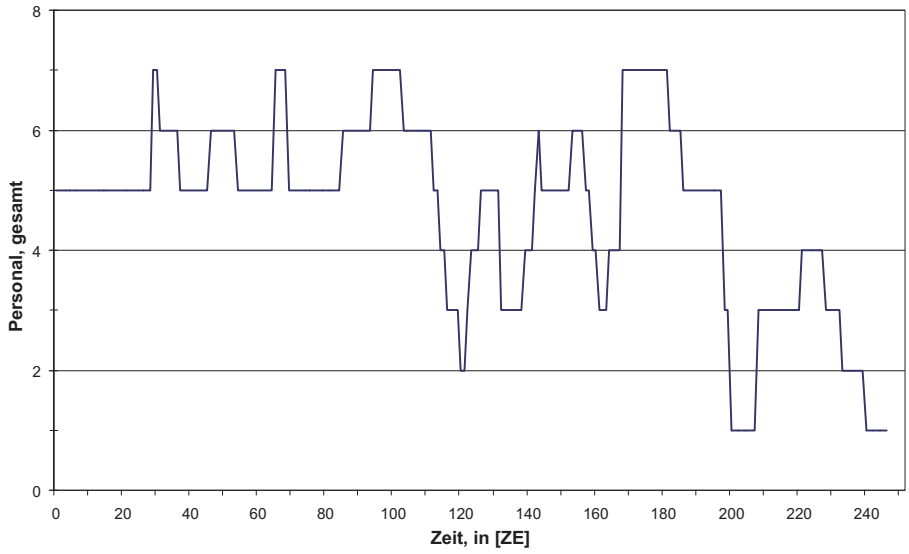


Abbildung 6.23.: Verteilung des Personaleinsatzes bei Begrenzung des Puffers auf $cp(b_{kp}) = 40$, $cp(b_{pv}) = \infty$, Beachtung Batch-Zuweisung. Dargestellt ist der Personalpool.

Personal-Pool

Das Personal wird zur Bearbeitung an den einzelnen Stufen benötigt, und stellt eine Möglichkeit dar, dynamisch auf Schwankungen im Auftragseingang, etc. zu reagieren. Hier soll untersucht werden, wie sich der Zielfunktionswert und die Laufzeit bei Variation des zur Verfügung stehenden Personals ändert. Für die drei Szenarien wurde die Kapazität $cp(h)$ systematisch bis auf zwei verkleinert. Die Kapazität der übrigen Ressourcen wurde nicht verändert. Die Ergebnisse sind in Tablle 6.5 dargestellt. Interessant ist, daß sich der Zielfunktionswert erst bei deutlicher geringerer Kapazität des Personal-Pools ändert. Dieser Effekt läßt vermuten, daß sich der Engpaß abhängig von der Zeit verschiebt, und somit eine Kombination aus Ressourcen und der Pool-Ressource darstellt.

Die Verringerung der Kapazität $cp(h)$ des Personal-Pools ist in den folgenden Abbildungen dargestellt, jeweils für die Fälle Beachtung bzw. Nicht-Beachtung der Batch-Zuweisung. Die Beschränkung des Personalpools auf $cp(h) = 5$ hat keinen Einfluß auf die Lösung, die Bestandsverläufe der Puffer und des Personalpools gleichen den Abbildungen 6.11 und 6.13.

Wird der Personalpool weiter verkleinert, so ergeben sich interessante Bestandsverläufe, die in Abbildung 6.24 bis 6.28 dargestellt sind. Deutlich ist die Fertig-

		MIN L_{max}						MIN C_{max}		
		Fall 1			Fall 2			(Fall 2)		
Szen.	$cp(h)$	L_{max}	C_{max}	t_{cpu}	L_{max}	C_{max}	t_{cpu}	L_{max}	C_{max}	t_{cpu}
1	5	-7	90	3	-7	90	1	0	90	1
1	3	-7	90	3	-7	90	2	0	90	1
1	2	0	96	3	-2	98	2	1	92	1
2	5	-90	291	*4*	-90	291	*5*	-59	291	*5*
2	3	-90	291	*4*	-90	291	*5*	-59	291	*2*
2	2	-71	291	*4*	-72	317	*44*	-38	293	*24*
3	10	-104	242	6	-104	247	34	47	250	*8*
3	6	-104	242	6	-104	247	41	47	250	*8*
3	3	-104	309	7	-104	318	54	151	311	*13*
3	2	-3	448	6	-50	452	*73*	313	442	*8*

Tabelle 6.5.: Einfluß der Kapazität des Personal-Pools $cp(h)$ auf die Laufzeit und den Zielfunktionswert. Die Puffer sind nicht kapazitativ begrenzt, die Laufzeit zur Bestimmung der Lösung ist angegeben. *Kursiv* sind die Versuche gekennzeichnet, die nach der Laufzeitbegrenzung von $t = 7200$ $[sec]$ keine Verbesserung der Lösung erzielten bzw. nicht die Optimalität der gefundenen Lösung zeigen konnten.

stellung der Touren und der enthaltenen Aufträge zu erkennen. Hier wird der Unterschied von Beachtung zu Nicht-Beachtung der Batch-Zuweisung deutlich. In Abbildung 6.24 sinkt der Bestand des Puffers zwischen Kommissionier- und Packbereich nahezu auf Null, während in Abbildung 6.25 weiterhin ein gewisser Basisbestand vorhanden ist. Hier wurden die Artikel zur Komplettierung der Aufträge kommissioniert, da die Batch-Zuweisung beachtet wird. Da nicht alle Positionen in den Aufträgen verwendet werden, bleiben einige Positionen bis zur ihrer Verwendung im Puffer. Bei der Zielfunktion MIN C_{max} arbeiten anfangs nur die Kommissionierresourcen, am Ende werden hier die Aufträge fertiggestellt, C_{max} unterscheidet sich allerdings nur marginal.

Die dargestellten Verläufe des Personal-Pools entsprechen den zuvor beschriebenen Tendenzen. Die Tatsache, daß bei einer Beschränkung des Personalpools auf $cp(h) = 2$ bis zu vier Kommissionierressourcen belegt sein können, ist durch die Übergangszeit zwischen zwei Positionen begründet. In dieser Zeit kann eine andere Palette (die sich an dem gleichen Arbeitsplatz befindet) bearbeitet werden. Bei den Pack- und Versandressourcen wurden keine Übergangszeiten berücksichtigt, hier tritt demnach auch dieser Effekt nicht auf. Das Personal ist über die gesamte Laufzeit voll ausgelastet, stellt in diesem Fall also den (erwarteten) Engpaß dar.

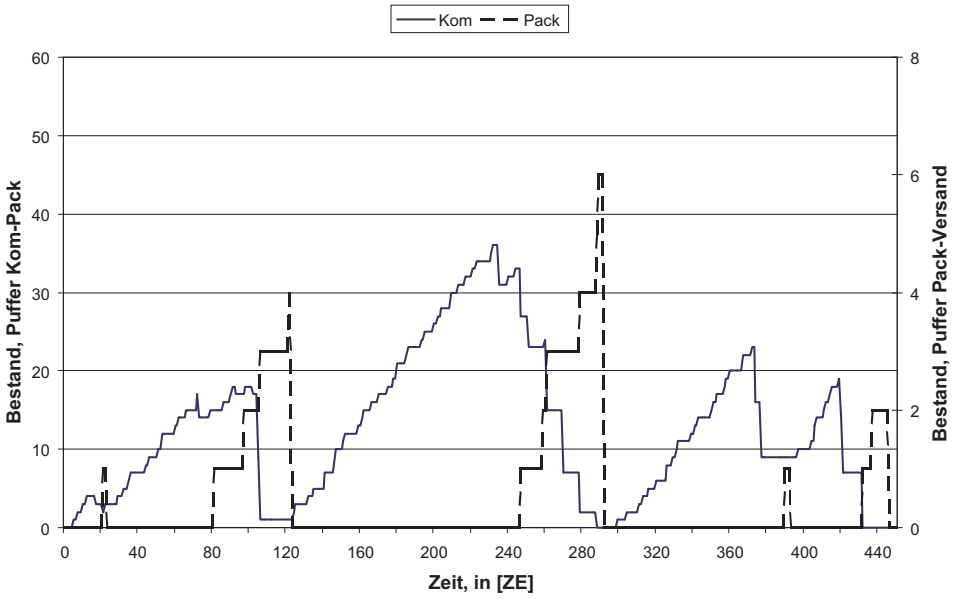


Abbildung 6.24.: Verlauf des Pufferbestandes, zugrunde liegt Szenario 3, Begrenzung des Personalpools auf $cp(h) = 2$, keine Beachtung der Batch-Zuweisung.

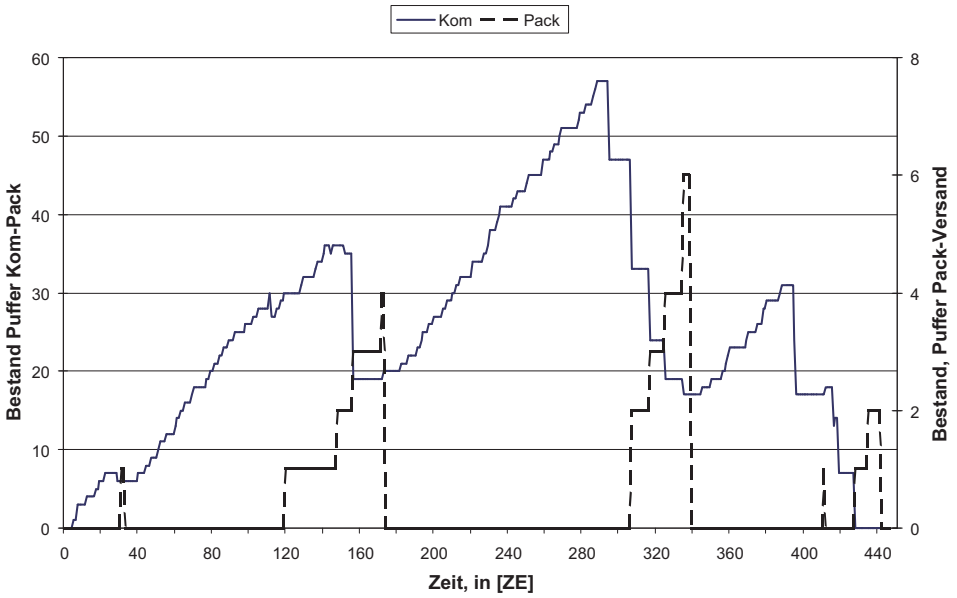


Abbildung 6.25.: Verlauf des Pufferbestandes, zugrunde liegt Szenario 3, Begrenzung des Personalpools auf $cp(h) = 2$, Beachtung der Batch-Zuweisung.

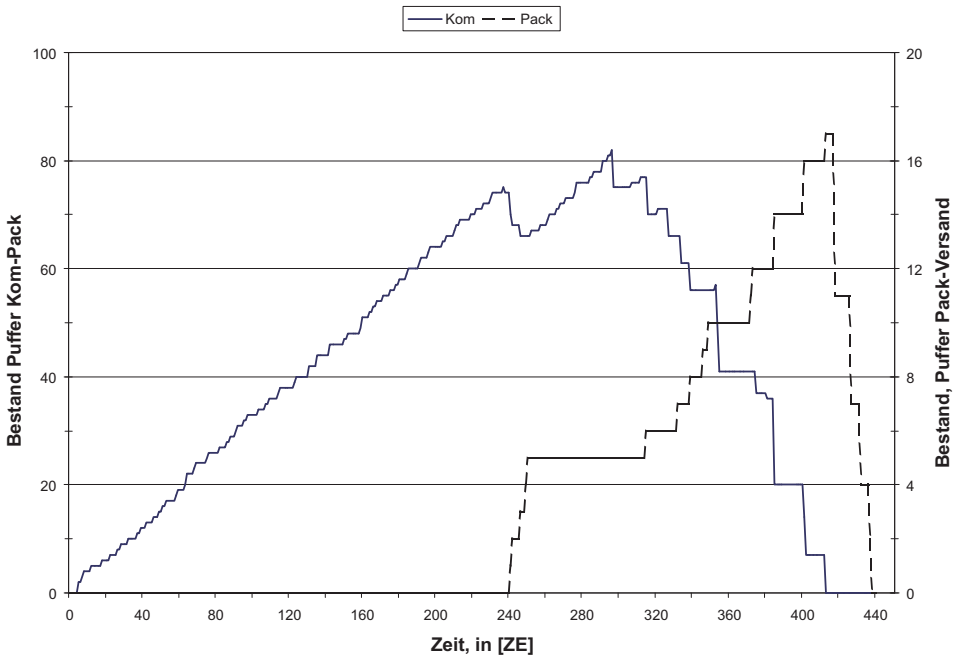


Abbildung 6.26.: Verlauf des Pufferbestandes, zugrunde liegt Szenario 3, Begrenzung des Personalpools auf $cp(h) = 2$, keine Beachtung der Batch-Zuweisung, Ziel MIN C_{max}.

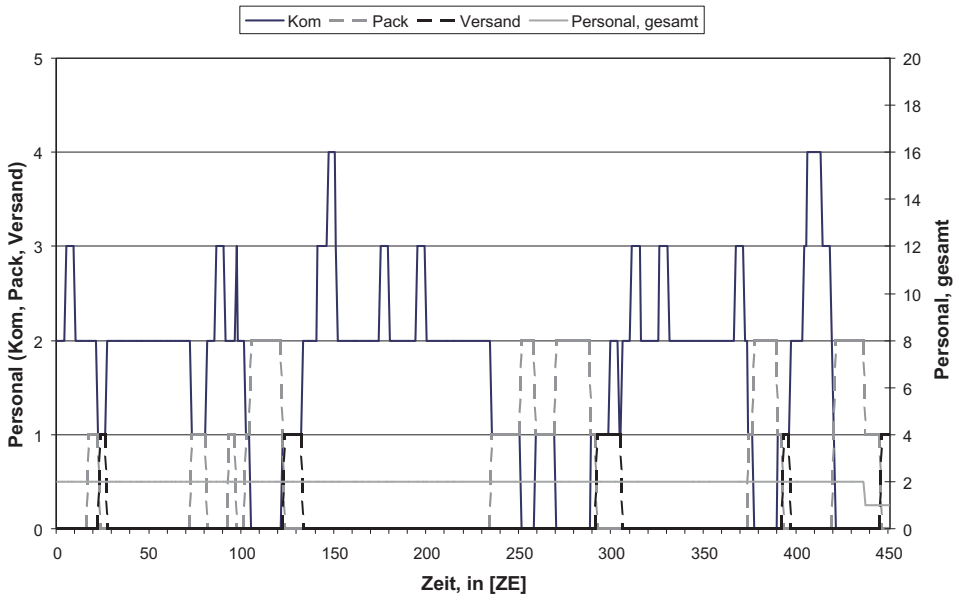


Abbildung 6.27.: Nutzung des Personal-Pools, zugrunde liegt Szenario 3, Begrenzung des Personalpools auf $cp(h) = 2$, keine Beachtung der Batch-Zuweisung.

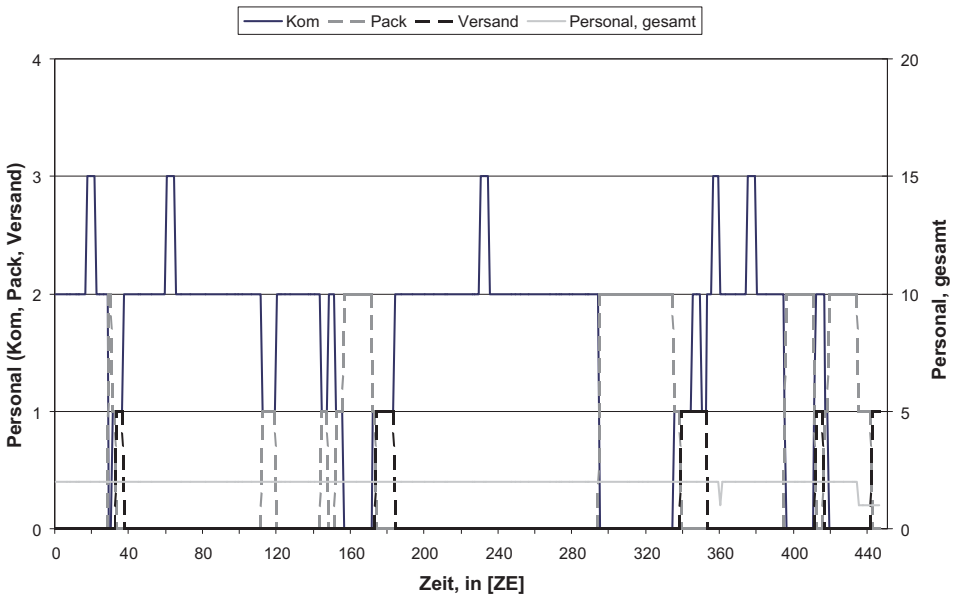


Abbildung 6.28.: Nutzung des Personal-Pools, zugrunde liegt Szenario 3, Begrenzung des Personalpools auf $cp(h) = 2$, Beachtung der Batch-Zuweisung.

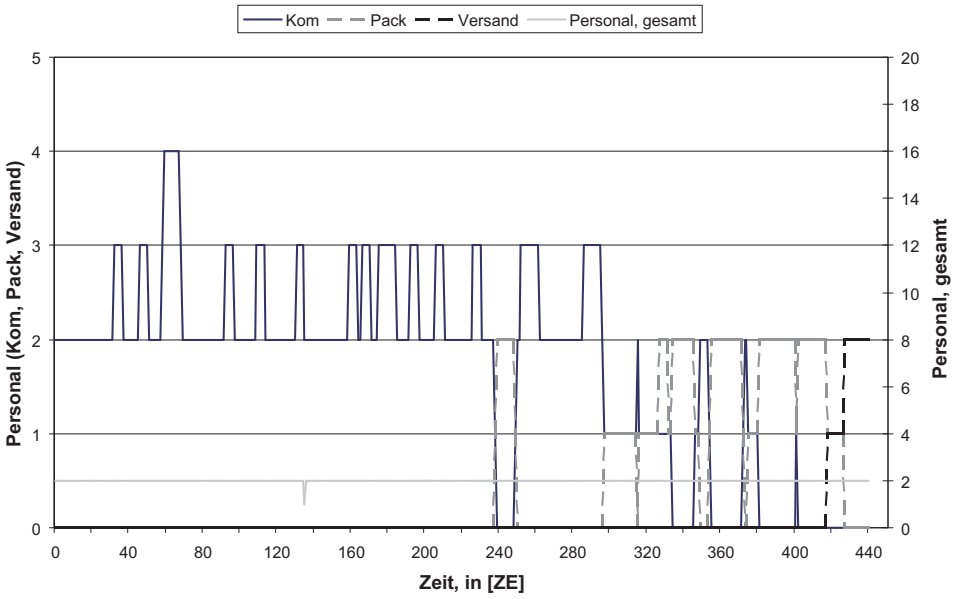


Abbildung 6.29.: Nutzung des Personal-Pools, zugrunde liegt Szenario 3, Begrenzung des Personalpools auf $cp(h) = 2$, keine Beachtung der Batch-Zuweisung, Ziel MIN C_{max}.

Laufzeitverhalten - Problemgröße

Interessant für eine praktische Anwendung ist das Laufzeitverhalten bei großen Problemen. Hier wurde anhand von zufällig generierten Szenarien (nach dem in Anhang B.1 dargestellten Algorithmus) untersucht, wie sich die Laufzeit bei Änderung der Problemgröße verhält. Die Puffer sind nicht kapazitativ beschränkt, der Personalpool ist beschränkt. Das Ergebnis ist in Abbildung 6.30 dargestellt. Hier ist die Laufzeit über der Anzahl der Operationen aufgetragen, die sich aus den zufällig generierten Szenarien ergeben. Interessant ist, daß die beste Lösung schnell gefunden wird, aber der Beweis der Optimalität bei Problemen mit mehr als 70 Operationen im Rahmen der Laufzeitbeschränkung nicht mehr geleistet werden konnte. Das läßt auf einen ausgeprägten Phasen-Übergang schließen.

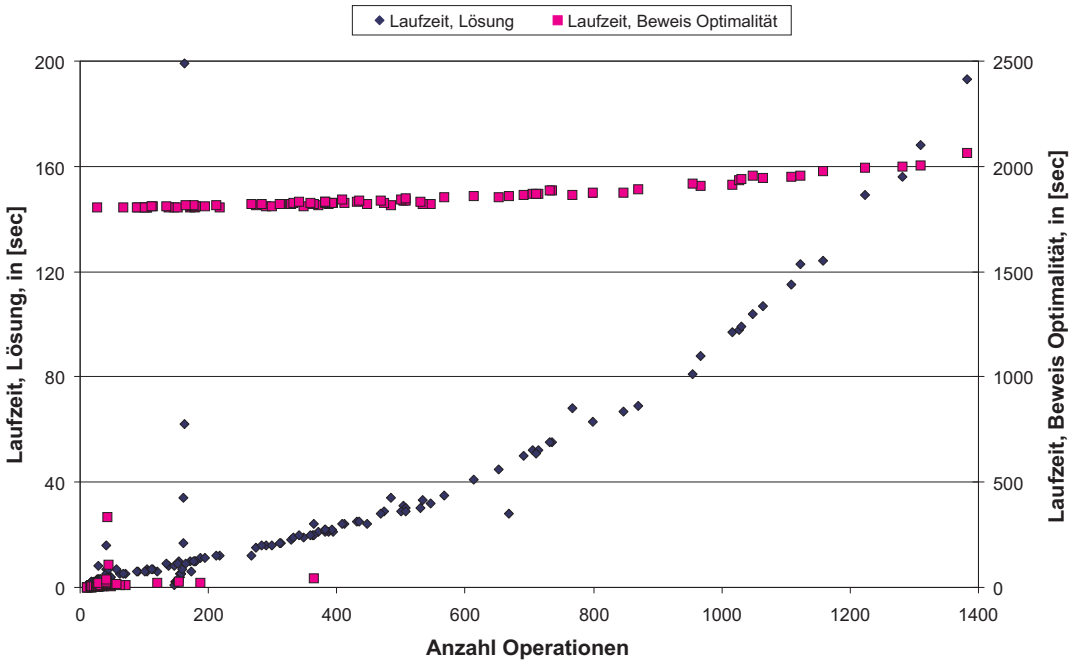


Abbildung 6.30.: Laufzeit in Abhängigkeit von der Anzahl der Operationen. Gezeigt ist die Laufzeit zur Bestimmung der besten Lösung und die Zeit, um die Optimalität zu beweisen. Bei Laufzeit-Werten $t_{cpu} > 1800$ $[sec]$ wurde die Laufzeitbeschränkung erreicht, die Optimalität konnte nicht bewiesen werden.

Laufzeitverhalten - Phasenübergang

Die Abhängigkeit der Laufzeit von dem Zielfunktionswert wurde für ein zufällig generiertes Szenario mit 20 Artikeln, 3 Touren, max. 6 Aufträgen pro Tour, max. 4 Positionen pro Auftrag und $cp(k) = 2$, $cp(p) = 2$, $cp(v) = 1$, $cp(b_{kp}) = cp(b_{pv}) = cp(h) = \infty$ und das vorgestellte Szenario 2 untersucht. Gleichung (6.22) gibt den zu erreichenden Zielfunktionswert L_{max} vor, d.h. das zugrunde liegende Constraint Satisfaction Problem wird gelöst. Die Laufzeiten für den Phasen-Übergang von zulässigen zu unzulässigen Lösungen sind in Abbildung 6.31 für ein zufällig erzeugtes Szenario dargestellt. Bei dem Übergang von den zulässigen zu den unzulässigen Lösungen ist ein deutlicher Anstieg in der Laufzeit zu erkennen. In einer praktischen Anwendung sollte es aus Laufzeitgründen vermieden werden, diesen Übergang zu untersuchen.

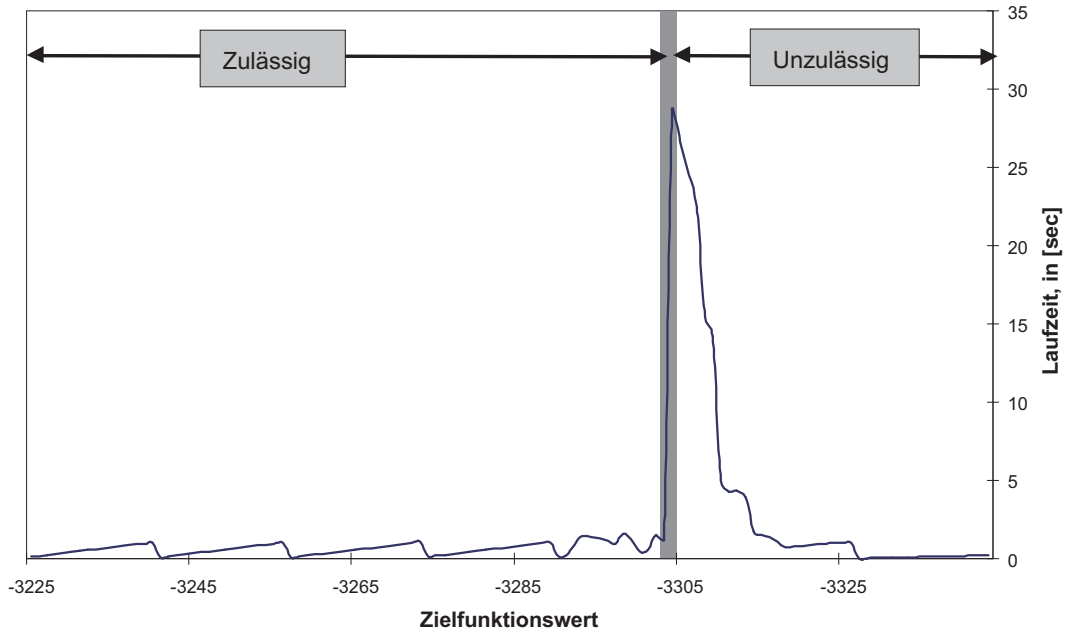


Abbildung 6.31.: Laufzeitverhalten des Phasen-Überganges von zulässigen zu unzulässigen Lösungen in Abhängigkeit von dem Zielfunktionswert.

Laufzeitverhalten - Suchverfahren

Die Laufzeit ist unter anderem von dem verwendeten Suchverfahren abhängig. Bei den bislang beschriebenen Ergebnissen wurde die binäre Suche, wie in Kapitel 5.5

beschrieben, verwendet. Hier wurde ebenfalls ein einfaches Verfahren beschrieben, das den Optimalwert systematisch von H annähert. Das erste nicht lösbare CSP zeigt, daß im letzten Schritt die optimale Lösung gefunden wurde. In Abbildung 6.32 werden die beiden Suchverfahren gegenübergestellt. Hier sind die Laufzeit und die Werte der Zielfunktion über der Anzahl der Operationen (und damit der Größe des Problems) abgetragen. Die Szenarien wurden zufällig nach dem in Anhang B.1 dargestellten Algorithmus erzeugt. Bei kleinen Problemen ist die einfache Suche der binären deutlich überlegen, bei 89 Operationen um den Faktor 29, bei 110 Operationen sogar um den Faktor 67. Bei einer größeren Anzahl an Operationen kehrt sich dieses Verhältnis jedoch um, bei 283 Operationen ist die binäre Suche um den Faktor 165 schneller als die einfache Suche.

Eine allgemeine Aussage kann nicht getroffen werden. Da ein Ziel der Untersuchung die Beherrschung von großen Problemen war, sollte hier die binäre Suche vorgezogen werden.

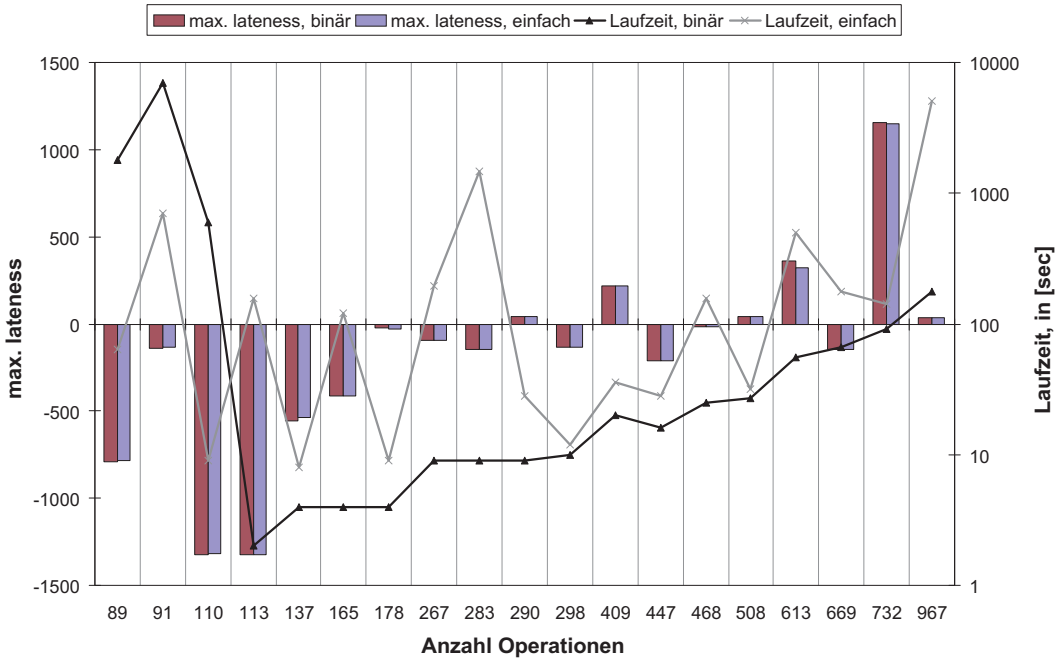


Abbildung 6.32.: Laufzeitverhalten in Abhängigkeit von dem verwendeten Suchverfahren, unterschieden in binäre Suche und einfache Suche, d.h. Annäherung von oben. Dargestellt ist auch der entsprechende Zielfunktionswert L_{max}, verwendet wurden zufällig generierte Szenarien.

In Abbildung 6.33 ist die Anzahl der Entscheidungspunkte für die binäre und einfache Suche in Abhängigkeit von den Operationen dargestellt. An einem Ent-

scheidungspunkt wird die nächste Variable ausgewählt. Beim idealen Fall einer Suche ohne backtrack-Schritt entspricht die Anzahl der Entscheidungspunkte der Anzahl der Variablen des COP. Für die in Abbildung 6.33 dargestellten Fälle konnte das Optimum in keinem Fall gezeigt werden, die Optimierung brach jeweils bei Erreichen der Laufzeitbeschränkung von $t = 7200$ [sec] ab. Bei größeren Problemen nimmt die Anzahl der Entscheidungspunkte ab, hier ist ein höherer Aufwand für die Prüfung und Sicherstellung der Konsistenz nötig. Die Anzahl der Variablen und Constraints steigt nahezu linear mit der Anzahl der Operationen. Jede Operation ist mindestens in einem Reihenfolge-Constraint enthalten. Weiterhin wird für jede Operation-Ressourcen-Kombination ein Kapazitäts-Constraint definiert. Die Beachtung der zusätzlichen Nebenbedingungen wie Übergangszeiten und die Transportzeit bei Nicht-Beachtung der Batch-Zuweisung führen zu disjunkten Constraints, die für alle Kombinationen von Operationen auf der entsprechenden Ressource aufgestellt werden. Die Definition von zusätzlichen Constraints führt weiterhin zu einem Anstieg der Anzahl Constraints.

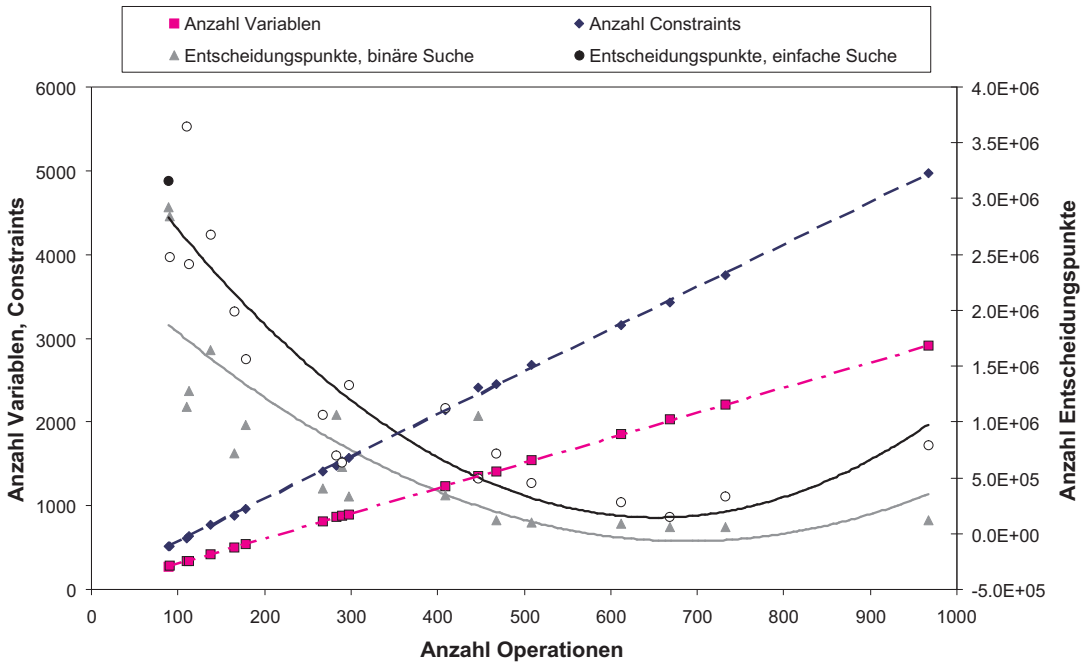


Abbildung 6.33.: Anzahl der Entscheidungspunkte für binäre und einfache Suche und Anzahl der Variablen sowie Constraints in Abhängigkeit von der Anzahl Operationen.

6.5.2. Bewertung

Es wurde gezeigt, daß die Berücksichtigung von praxisrelevanten Anforderungen möglich ist. Die Rechenzeit, um die Anwendungsfälle zu lösen, ist hinreichend gering, somit ist eine Anwendung in einem praktischen Umfeld möglich. Die Untersuchungen zum Phasen-Übergang machen deutlich, daß aus Laufzeitgründen nicht die Bestimmung der optimalen, sondern einer sehr guten Lösung im Vordergrund stehen sollte.

Es wurde aufgezeigt, wie das Personal auf die einzelnen Stufen verteilt werden kann, um die geforderten Fertigstellungszeitpunkte einzuhalten. Hier ist allerdings der (örtliche) Wechsel eines Mitarbeiters von einem Bereich zu einem anderen noch nicht berücksichtigt, was an dem zackenartigen Verlauf der Personal-Grafiken zu erkennen ist. Dies ließe sich über eine Glättung in einem zweiten Schritt integrieren. Wird der Ansatz über einen längeren Zeitraum (beispielsweise mit Vergangenheitsdaten) angewendet, können Aussagen für eine längerfristige Flexibilisierung des Personaleinsatzes getroffen werden.

Für die Zielfunktionswerte ist die Reihenfolge der Positionen entscheidend, die Beachtung des Batches hat erstaunlicherweise nur einen kleinen Einfluß. Das kann an der Struktur der zufällig erzeugten Probleme liegen.

Die Begrenzung des Puffers hat einen deutlichen Anstieg der Laufzeit zur Folge. Die eingesetzten Verfahren zur Auswahl der Variablen berücksichtigen den Füllgrad des Puffers nur indirekt, somit kann es zum Effekt des trashing kommen. Ist der Puffer voll belegt, können Operationen ausgewählt werden, die eine Einlagerung bewirken. Da anschließend Kapazitäts-Constraints verletzt werden, wird die Zuweisung rückgängig gemacht und eine neue Variable gesucht. Die Objektstruktur bewirkt, daß deutlich mehr Operationen einen Zugang als einen Abgang des Puffers bewirken. Hier ist durchaus noch Potential für Verbesserungen der Variablen-Sortierung. In der Realität wird der Puffer nie einen so deutlichen Engpaß wie auf Seite 84 darstellen, andernfalls sollte die Planung, d.h. die Konfiguration des Systems, überdacht werden.

Die Laufzeit für Anwendungen ohne Pufferbeschränkung zeigt die Anwendbarkeit der Ansätze, im Bereich der kurzfristigen Entscheidungen (d.h. im Sekundenbereich) sind die eingesetzten Verfahren allerdings zu langsam. Hier zeigt sich, daß die Grenzen der der COP Ansätze in der Online-Fähigkeit liegt.

Die Ergebnisse sind vielversprechend, mögliche Erweiterungen liegen in der Verbesserung der Suche über Weiterentwicklung der vorgestellten Variablen-Sortierverfahren. Weiterhin ist eine feinere Differenzierung der Constraints, die die Zielfunktion beschreiben, anzustreben. So wäre es bei einer praktischen Anwendung interessant, zuerst die Zulässigkeit der Fertigstellungszeitpunkte zu zei-

gen (beispielsweise zur Bestätigung der Bestellungen), um anschließend das eingesetzte Personal zu minimieren. Hier könnten auch Szenarien mit flexiblen Einsatzkräften, etc. durchgespielt werden.

6.6. Zusammenfassung

Die vorgestellte Modellierung und Formulierung als Constraint Optimization Problem wurde in diesem Kapitel auf mehrstufige Kommissioniersysteme angewendet. Zunächst wurde eine allgemeine Darstellung gegeben, um auf die Probleme hinzuführen. Hier sind vor allem die (begrenzten) Puffer zwischen den Stufen und die über den Tag wandernden Belastungsspitzen interessant. Diese führen zu wandernden Engpässen und stellen Anforderungen für den Einsatz von flexiblen Ressourcen. Weiterhin wurden die zu erwartenden Effekte bei Beachtung (Fall 1), bzw. Nicht-Beachtung (Fall 2) einer Batchzuweisung dargestellt.

Das Problem wurde als Constraint Satisfaction Problem modelliert. Die in Kapitel 4 entwickelten Ansätze wurden erweitert, um die spezielle Unterscheidung von Fall 1 und Fall 2 zu integrieren. Das Hinzufügen der Zielfunktion als Constraint überführt das CSP in ein COP.

Die Effekte der Beschränkung der Puffer-Kapazität und des Personal-Pools wurden mit drei Szenarien und zufällig erzeugten Problemen untersucht, die Ergebnisse wurden diskutiert. Die Einflüsse der Beachtung der Batch-Zuweisung zu Nicht-Beachtung der Batchzuweisung fallen erstaunlich gering aus, bei einigen zufällig erzeugten Szenarien konnten keine Unterschiede festgestellt werden. Dieser Effekt kann auf die Struktur der generierten Touren, Aufträge und Positionen zurückgeführt werden. Die Laufzeiten zur Bestimmung einer guten Lösung waren für einen praktischen Einsatz akzeptabel, der Beweis der Optimalität konnte aus Laufzeitgründen nur für die kleineren Probleme erbracht werden. Eine genauere Betrachtung des Laufzeitverhaltens schließt das Kapitel ab.

7. Der COP-Ansatz für ein Umschlagsystem Schiene-Schiene

Das Transportvolumen des Güterverkehrs auf der Schiene stagniert. Große Kunden wie die Deutsche Post AG haben sich gegen den Transport auf der Schiene entschieden. Ein Grund liegt in der geringen Geschwindigkeit des Transportes auf der Schiene. In einem Vergleich Straße zu Schiene wurde festgestellt, daß ein Container, der auf der Schiene quer durch Deutschland (Nord-Süd) transportiert wurde, eine durchschnittliche Geschwindigkeit von $v = 7\,[\frac{km}{h}]$ erreichte[1]. Dieses schlechte Ergebnis kommt unter anderem durch den großen Aufwand bei der Zugauflösung und anschließenden Zugbildung, d.h. dem Umrangieren in den Terminals, zustande. Im intermodalen Verkehr sollen eine engere Kopplung der unterschiedlichen Verkehrsträger Straße, Schiene und Binnenschifffahrt sowie neue flexible Terminalkonzepte den Gütertransport wieder konkurrenzfähig machen. Die kurzen Entfernungen, um das Terminal mit den Kunden im sog. Hinterland zu verbinden, bedienen sehr flexible Lkw. Auf den großen Distanzen, d.h. zwischen den Terminals, operieren sinnvollerweise Güterzüge (N.N. (1995)).

Vom BMBF[2] wurde zeitweise ein Projekt gefördert, in dem die Deutsche Bahn AG und Anbieter von Krananlagen unterschiedliche Konzepte und Produktionsformen entwickelten, um den Güterverkehr flexibler, kostengünstiger und damit wieder attraktiv zu machen N.N. (1997). Die Ansätze sind Gegenstand zahlreicher Publikationen und Tagungen, siehe beispielsweise Alicke und Arnold (1998), Alicke, Arnold, Nienhaus und Franke (1999), Arnold und Rall (1996a), Arnold und Rall (1996b). Die Entfernung, ab der ein Güterzug wirtschaftlich eingesetzt werden kann, liegt bei $d = 400\,[km]$. Diese Entfernung soll mit den neuen Terminalkonzepten verringert werden. Um einen Umschlag schnell und flexibel zu gestalten, steigen die Anforderungen an die neuen Terminals.

Von der Firma Noell wurde die Variante der sog. Mega-Drehscheibe oder des

[1] Der Verkehrsminister F. Müntefering sprach anläßlich der EUROCARGO von $v = 18\,[\frac{km}{h}]$, siehe N.N. (1999).

[2] Bundesministerium für Bildung und Forschung.

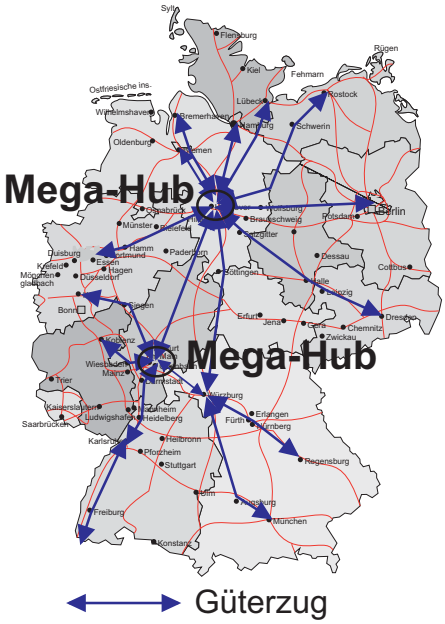


Abbildung 7.1.: Einsatz von Güterzügen im kombinierten Ladungsverkehr.

Mega Hubs entwickelt. Hier wird die starre Kopplung Ladeeinheit-Tragwagen aufgehoben, der damit verbundene Rangieraufwand entfällt. Im Konzept Mega Hub steigen die Ladeeinheiten mit Hilfe von Kränen und einer Sortieranlage von einem Zug auf einen anderen um. Die Aufenthaltszeiten (der Züge und damit Ladeeinheiten) im Terminal können wesentlich reduziert werden, siehe Franke und Häffner (1996), Franke (1997). Die Züge operieren nach einem festgelegten Fahrplan, daher haben Verspätungen ähnliche Auswirkungen auf das Gesamtnetz wie im Personenverkehr und sind zu vermeiden.

Der Mega Hub zeichnet sich durch einen modularen Aufbau und damit verbundene Skalierbarkeit aus. Wegen der relativ komplexen Struktur ist die Steuerung jedoch nicht trivial. Die Optimierungspotentiale liegen in einem reibungslosen und vor allem kostenminimalen Ablauf. Das Layout einer Modellanlage ist in Abbildung 7.2, die Integration in das Gesamtnetz exemplarisch in Abbildung 7.1 dargestellt.

In Kapitel 7.1 wird das untersuchte Terminal beschrieben, und die Ziele des Modells werden umrissen. Die Modellierung als mehrstufiges Umschlagsystem schließt sich in Kapitel 7.2 an. Hier werden die umzuschlagenden Ladeeinheiten klassifiziert, um sie mit den in Kapitel 4 dargestellten Ansätzen abzubilden. Weiterhin wird kurz auf die Berechnung der Umschlagszeiten und die Modellierung von überlappenden, d.h. alternativen Einsatzbereichen für Kräne, eingegangen.

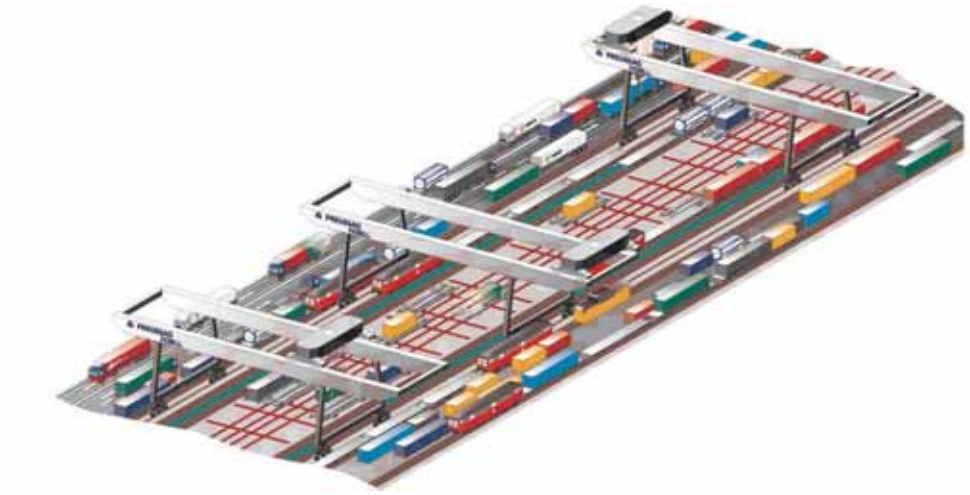


Abbildung 7.2.: Modell des Terminals Mega Hub, dargestellt sind die Gleise, drei Portal-Kräne, die Sortieranlage und die straßenseitige Anbindung.

Diese Modellierung wird in Kapitel 7.3 genutzt, um ein Constraint Optimization Problem zu formulieren. Anschließend wird in Kapitel 7.4 die Zuordnung der Gleise zu den einfahrenden Zügen untersucht. Es wird ein gemischt-ganzzahliges Optimierungsmodell formuliert, das exakt gelöst werden kann. In Kapitel 7.5 schließt sich die Anwendung mit konkreten Ergebnissen und einer Diskussion an.

7.1. Terminal Mega Hub

Das Prinzip-Layout des Terminals ist in Abbildung 7.3 dargestellt. Die sechs Gleise werden von maximal zwölf Kränen überspannt, deren Einsatzbereiche sich überlappen können. Theoretisch kann ein Kran den gesamten Umschlagsbereich abfahren, somit kann ein Kran bei ungleichmäßiger Verteilung der Last dem angrenzenden Kran aushelfen. Die einzelnen Kranbereiche sind über eine Sortieranlage verbunden, in Abbildung 7.3 in der Mitte zwischen Gleis 3 und 4 dargestellt. Die Sortieranlage wird vollautomatisch betrieben und besteht aus einem System von parallel und rechtwinklig zueinander verlaufenden handelsüblichen Eisenbahnschienen. Zum Transport und zur Pufferung der Ladeeinheiten werden linearmotorgetriebene Rollpaletten eingesetzt, die zum besseren Verständnis im folgenden als fahrerlose Transport-Fahrzeuge (FTF) bezeichnet werden. Die Baugruppen für den Antrieb der Linearmotoren sind in der Fahrbahn integriert (Bauer (1998)). Auf der Anlage werden Container, Wechselbrücken und Sattelauf-

lieger umgeschlagen. Die Sortieranlage besteht aus zwei (Einbahn-)Gleisen zum Transport der Ladeeinheiten und aus Puffern am Rand.

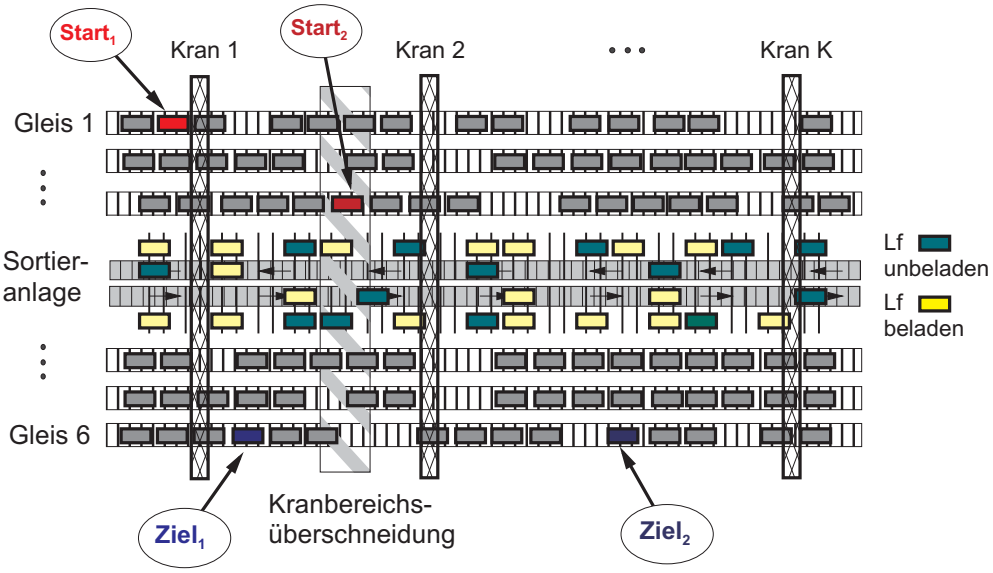


Abbildung 7.3.: Prinzip-Layout des Mega Hub.

Das Terminal soll einen *maximalen* Servicegrad (d.h. Häufigkeit der pünktlich umgeschlagenen Ladeeinheiten) gewährleisten. Um die Ladeeinheiten umzuschlagen, werden Ressourcen wie die Kräne, die FTF der Sortieranlage und Personal benötigt. Das vorrangige Ziel ist es, alle Ladeeinheiten gemäß der gegebenen Umschlagsmatrix umzuschlagen und die in einem Fahrplan gegebenen Abfahrtzeiten der Züge einzuhalten. In dem Terminal werden sogenannte *Bündel* von Zügen bearbeitet. Ein Bündel besteht aus sechs Zügen, die nacheinander einfahren, bei der prototypischen Anlage im Abstand von $\Delta t = 8\,[min]$. Das Ziel wird auf die Minimierung der maximalen Verspätung (lateness) zurückgeführt. Hierzu müssen die folgenden Fragestellungen gelöst werden:

- In welcher Reihenfolge sind die Ladeeinheiten umzuschlagen?
- Sollen die Kran-Bereiche disjunkt oder überlappend sein?
- Wie groß sollten die Kran-Bereiche sein?
- Welcher Kran soll eine Ladeeinheit des überlappenden Bereiches umschlagen?
- Auf welchen Gleisen sollen die Zügen einfahren?

Die Anlage kann manuell (durch Kranführer) oder automatisch betrieben werden. Beim Einsatz von automatischen Kränen muß sichergestellt werden, daß es während eines Umschlages nicht zu Kollisionen kommt. Ein Ansatz, der Prioritätsregeln anwendet, wurde von Meyer (1999) entwickelt und in einem Petri-Netz-basierten Simulator evaluiert. Das modellierte System zeigt deutlich die Komplexität des Terminals.

Zum Einsatz im praktischen Umfeld ist eine schnelle Planung von Umschlags-Reihenfolgen oder Belegungen unumgänglich. Die Zeit, die zur Lösung des Problems zur Verfügung steht, wird durch die Ankunftzeiten der Züge bestimmt. Falls eine Verspätung auftritt, kann im Extremfall die Umschlagsreihenfolge erst bei Ankunft des Zuges im Terminal geplant werden. In der Regel wird die Ankunft des Zuges nach Passieren der letzten Blockstrecke anvisiert, somit stehen ca. 5 min. zur Berechnung zur Verfügung. Als Eingangsdaten liegt der Fahrplan der Züge sowie eine Umschlagsmatrix vor. Hier ist für jede Ladeeinheit die Position auf dem ankommenden Zug und die Absetzposition auf dem Ziel-Zug gegeben. Die Positionen werden durch den Zuglauf und sich hieraus ergebenden Absetzgruppen bestimmt.

Das Modell muß also Fragen beantworten, ob der Fahrplan noch einzuhalten ist, bzw. welche Verspätungen einzukalkulieren sind. Weiter interessiert, welche Ladeeinheiten bei einem fixierten Fahrplan umgeschlagen werden können und welche auf den nächsten Zug in die Zielrichtung warten müssen. Sind die Züge nicht voll besetzt, interessieren Fragestellungen, wie viele Kräne für das Umschlagen der Ladeeinheiten unter Einhaltung der Fahrpläne minimal benötigt werden.

7.2. Modellierung

In Kapitel 4 wurde der Begriff des Systems eingeführt, um anschließend die Elemente und die Beziehungen untereinander darzustellen. Im folgenden werden die realen Elemente wie Kran, Ladeeinheit etc. auf eine abstrakte Ebene transformiert, um sie in das Modell zur Optimierung des Terminals Mega Hub zu integrieren.

7.2.1. Vorüberlegungen

Um eine Ladeeinheit von einer Startposition auf einem Tragwagen auf eine Zielposition umzuschlagen, werden Ressourcen wie Kran(e) und evtl. ein FTF des Sortierspeichers benötigt. Abhängig von der Start- und Zielposition läßt sich ein Umschlag wie folgt unterteilen (siehe auch Abbildung 7.4):

- *Direkter Umschlag:* Die Ladeeinheit wird nur einmal gehandhabt. Der Kran fährt leer zu der Start-Position, nimmt die Ladeeinheit auf, fährt beladen zu der Zielposition und gibt die Ladeeinheit ab. Anschließend wartet der Kran auf den nächsten Fahrauftrag. Der direkte Umschlag sollte bevorzugt verwendet werden.

- *Indirekter Umschlag:* Die Ladeeinheit wird mehrfach gehandhabt, von dem gleichen oder unterschiedlichen Kränen und evtl. von einem FTF des Sortierspeichers. Durch zeitliche (Fahrplan) oder/und örtliche (Start- und Zielposition) Restriktionen kann es nötig sein, die Ladeeinheiten zu puffern. Ein Kran fährt leer zu der Start-Position, nimmt die Ladeeinheit auf, fährt (auf kürzestem Weg) zu dem Sortierspeicher, wo ein leeres FTF wartet und lädt die Ladeeinheit auf diesem FTF ab. Das FTF puffert die Ladeeinheit (und transportiert sie evtl. in den Zielbereich), bis der Kran des Zielbereiches die Ladeeinheit aufnimmt, zu der Zielposition transportiert und absetzt.

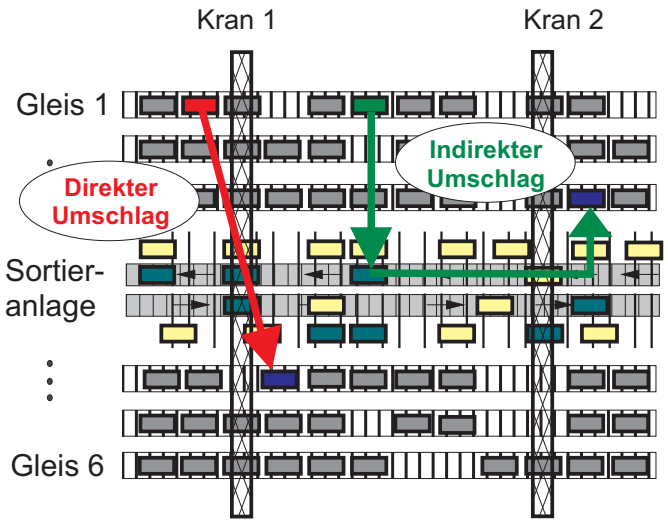


Abbildung 7.4.: Unterscheidung nach *direkten* und *indirekten* Umschlägen.

Verteilung Direkt-Indirektumschlag

Die Anzahl der *möglichen* Direktumschläge korreliert mit der Anzahl der eingesetzen Kräne. Die Verteilung der direkten/indirekten Umschläge und damit die Anzahl der Ladeeinheiten, die die Sortieranlage belasten, kann sehr einfach angegeben werden. Hierbei wird unterstellt, daß die Kranbereiche disjunkt sind.

Es wird angenommen, daß alle Züge gleich beladen und alle Gleise mit Zügen belegt sind. Die Züge stehen auf G Gleisen und werden von insgesamt K Kränen umgeschlagen. Insgesamt sollen C Ladeeinheiten umgeschlagen werden, eine Gleichverteilung der Umschlagsmatrix wird unterstellt. Die Wahrscheinlichkeiten, daß eine Ladeeinheit auf einen bestimmten Zielbereich umgeschlagen wird, lassen sich einfach bestimmen:

$$p_g = \frac{1}{G} \text{ - gleicher Zug} \tag{7.1}$$

$$p_d = \frac{1}{K} \text{ - direkter Umschlag (gleicher Kranbreich)} \tag{7.2}$$

$$p_i = \frac{K-1}{K} \text{ - indirekter Umschlag (unterschiedlicher Kranbereich)} \tag{7.3}$$

$$p_p = \frac{1}{C} \text{ - genau eine Position} \tag{7.4}$$

Da $p_d + p_i = 1$ gilt, läßt sich die Verteilung der direkten und indirekten Umschläge in Abhängigkeit von der Krananzahl K ermitteln. Das Ergebnis ist in Abbildung 7.5 dargestellt.

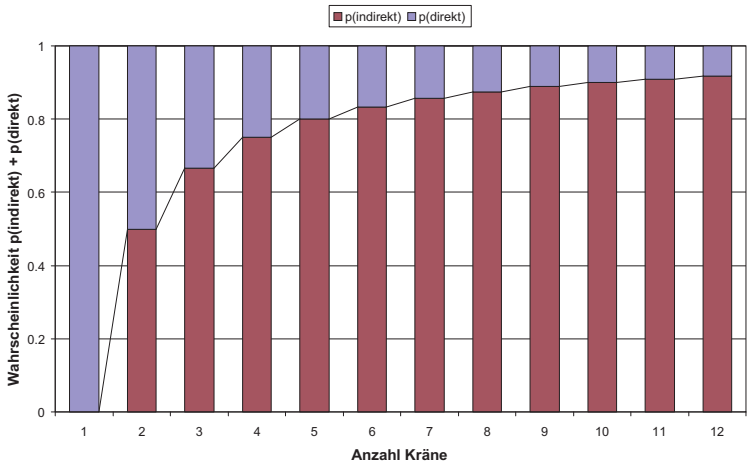


Abbildung 7.5.: Verteilung der *direkten* und *indirekten* Umschläge in Abhängigkeit der Anzahl eingesetzter Kräne.

Umschlagszeiten

Die Ressourcen werden durch kinematische und geometrische Daten beschrieben. Zur Bestimmung von optimalen Reihenfolgen ist die Dauer eines Umschlages interessant. Der Bewegungsvorgang eines Kranes unterteilt sich in die Kranfahrt parallel zu den Gleisen und die Katzfahrt[3] senkrecht zu den Gleisen. Die Bewegungen können überlagert werden und ähneln der Synchronfahrt von Regalbediengeräten. Nach einer gewissen Beschleunigungsphase wird eine Phase konstanter Geschwindigkeit erreicht, anschließend wird wieder abgebremst. Im folgenden wird die Anfahr- und Brems-Beschleunigung nach Arnold (1995) mit $a = \frac{2 \cdot |a_1 \cdot a_2|}{a_1 + |a_2|}$ approximiert[4].

Für die Fahrzeit t gilt somit mit der zurückzulegenden Strecke l

$$t = \begin{cases} \frac{l}{v} + \frac{v}{a} & \text{für } l \geq \frac{v^2}{a} \\ 2 \cdot \sqrt{\frac{l}{a}} & \text{für } l < \frac{v^2}{a} \end{cases} \tag{7.5}$$

Die Koordinaten der Start- und Zielposition sind $P_s = (x_s, y_s)$ bzw. $P_z = (x_z, y_z)$. Somit ist die zurückzulegende Strecke in Katz-Richtung $l_{katz} = |y_s - y_z|$ und in Kran-Richtung $l_{kran} = |x_s - x_z|$. Mit Gleichung (7.5) lassen sich die Fahrzeiten für die Katzfahrt t_{katz} und die Kranfahrt t_{kran} ermitteln. Die Dauer einer Fahrt wird durch $t^{fahr} = \text{MAX}\,[t_{kran}, t_{katz}]$ bestimmt.

Ein Umschlag setzt sich aus unterschiedlichen Teil-Vorgängen zusammen. Der Kran fährt leer zu der Start-Position, positioniert den Spreader[5], nimmt die Ladeeinheit auf, fährt voll zu der Ziel-Position und setzt die Ladeeinheit nach erneuter Positionierung auf der Ziel-Position ab. Die benötigte Zeit setzt sich also aus Leerfahrt t^{leer}, Aufnehmen[6] t^{auf}, Vollfahrt t^{voll} und Absetzen t^{ab} zusammen. Die gesamte Dauer eines Umschlag-Vorganges ergibt sich zu

$$t = t^{leer} + t^{auf} + t^{voll} + t^{ab} \tag{7.6}$$

Die Zeiten für das Aufnehmen und Absetzen sind bei Ladeeinheiten des gleichen Typs gleich, die Dauer der Vollfahrten hängt von den Start- und Zielpositionen

[3] Die hier verwendete *Drehkatze* der Portalkräne kombiniert in dem Maschinenhaus das Hubwerk und das (Katz-)Fahrwerk und erlaubt die Drehung der Ladeeinheit um bis zu 180^o.

[4] Hierbei bezeichnet a_1 die Anfahr- und a_2 die Bremsbeschleunigung.

[5] Spreader: Lastaufnahmemittel des Kranes, für Container, Wechselbrücken und Sattelauflieger geeignet. Kann in der Länge an die Containerlänge angepaßt werden, zur Aufnahme von Wechselbrücken mit Greifzangen ausgestattet.

[6] Aufnehmen bedeutet das Absenken des Spreaders, Positionieren über der Ladeeinheit, Eingreifen des Spreaders in die Eckbeschläge bzw. der Greifzangen bei Wechselbehältern und Sattelaufliegern und Anheben der Ladeeinheit; entsprechendes gilt für das Absetzen der Ladeeinheiten.

ab, läßt sich also ebenfalls a-priori bestimmen. Die Leerfahrten sind von der aktuellen Position des Kranes abhängig, und damit von der zuletzt umgeschlagenen Ladeeinheit, sind also reihenfolgeabhängig (siehe Abbildung 4.6).

Bei dem Umschlag von Containern greift der Spreader in die Eckbeschläge ein; dazu muß er vor dem Aufnehmen auf die entsprechende Größe des Containers (20', 40', etc.) eingestellt werden. An dem Spreader sind zum Umschlag von Wechselbehältern und Sattelaufliegern ausfahrbare Greifzangen angebracht. Auch hier entsteht eine zusätzliche Zeit zur Vorbereitung des Spreaders, die der Leerfahrzeit zugerechnet wird (vgl. Kapitel 7.3.4).

7.2.2. Klassifizierung der Umschläge

Um für das Mega Hub Problem die in Kapitel 4 vorgestellte Modellierung anzuwenden, werden *Operationen* identifiziert, die auf *Ressourcen* einzuplanen sind. Eine Operation sollte ohne Unterbrechung von einer Ressource ausgeführt werden. Ein Auftrag besteht häufig aus mehreren Operationen. Von der Wahl der Operationen hängt die Komplexität des Modells und damit die nötige Rechenzeit ab. Die Modellierung jedes Teilvorganges als Operation erscheint wenig sinnvoll, daher wurde ein Umschlag-Vorgang, der sich aus den vier beschriebenen Teil-Vorgängen zusammensetzt, als eine Operation definiert. Der Umschlag einer Ladeeinheit besteht bei einem direkten Umschlag aus einer Operation (Vorgang), bei einem indirekten aus zwei bzw. drei Operationen (Vorgängen), abhängig davon, ob der Sortierspeicher genutzt wird. Diese Operationen definieren einen Auftrag.

Es ergeben sich also Aufträge, die aus einer, zwei oder drei Operationen bestehen, wobei jede Operation vier Teil-Vorgänge enthält. Abhängig von der Start- und der Zielposition der Ladeeinheiten lassen sich nun drei unterschiedliche Umschlags (Auftrags)-Kategorien identifizieren, die im Modell durch eine unterschiedliche Anzahl von Operationen dargestellt werden. Bei dieser Betrachtung wird das Terminal in K gleich große Kranbereiche unterteilt. Die Positionen können in den gleichen oder unterschiedlichen Kranbereichen und oberhalb oder unterhalb des Sortierspeichers liegen. Die drei Kategorien sind:

- *Kategorie 1 (F_1):* Start- und Zielposition der umzuschlagenden Ladeeinheit liegen in dem gleichen Kranbereich und auf der gleichen Seite des Sortierspeichers, siehe Abbildung 7.6. Wenn sich die Aufenthaltszeiten des Start- und des Zielzuges überlappen, ist ein direkter Umschlag möglich. Die Pufferung der Ladeeinheit auf dem Sortierspeicher ist nur sinnvoll, wenn sich die Aufenthaltszeiten nicht überlappen, ansonsten würden unnötige We-

ge zurückgelegt. Ein Auftrag dieser Kategorie wird durch *eine* Operation Start-Position → Ziel-Position modelliert.

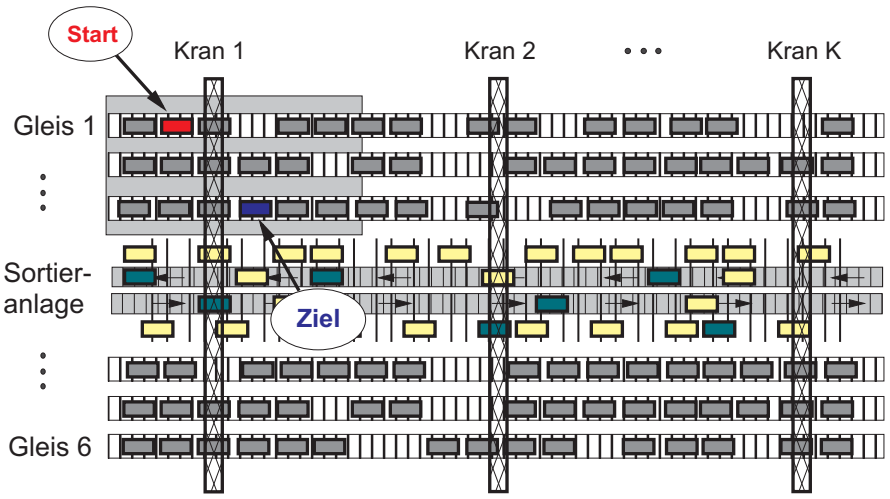


Abbildung 7.6.: Umschlag nach Kategorie 1.

- *Kategorie 2 (F_2):* Liegen die Start- und Zielposition in dem gleichen Kranbereich, aber auf unterschiedlichen Seiten der Sortieranlage (Abbildung 7.7), ist ein direkter Umschlag möglich. Durch zeitliche Constraints (z.B. Ausfahrzeit des Start-Zuges) kann es sinnvoll sein, die Ladeeinheit auf der Sortieranlage zu puffern und den Umschlag zu einem späteren Zeitpunkt zu beenden. Es wird angenommen, daß die Ladeeinheit auf dem Verbindungsweg der Start- zu der Zielposition gepuffert werden kann, somit ist kein zusätzlicher Weg, sondern nur Zeit für das Absetzen, eine Leerfahrt und das anschließende Aufnehmen der Ladeeinheit nötig. Dieser Auftrag wird durch *zwei* Operationen Start-Position → Sortierspeicher und Sortierspeicher → Ziel-Position modelliert. Diese Kategorie gilt auch für Umschläge der Kategorie 1, wenn sich die Aufenthaltszeiten der ein- und ausfahrenden Züge nicht überlappen.

- *Kategorie 3 (F_3):* Für den Fall, daß die Start- und Ziel-Position in unterschiedlichen Kranbereichen liegen (siehe Abbildung 7.8) ist in jedem Fall ein indirekter Umschlag nötig. Hierbei spielt es keine Rolle, ob die Aufenthaltszeiten der Züge überlappen oder nicht. Der erzeugte Auftrag besteht aus drei Operationen Start-Position → Sortierspeicher(Start-Bereich), Sortierspeicher(Start-Bereich) → Sortierspeicher(Ziel-Bereich) und Sortierspeicher(Ziel-Bereich) → Ziel-Position.

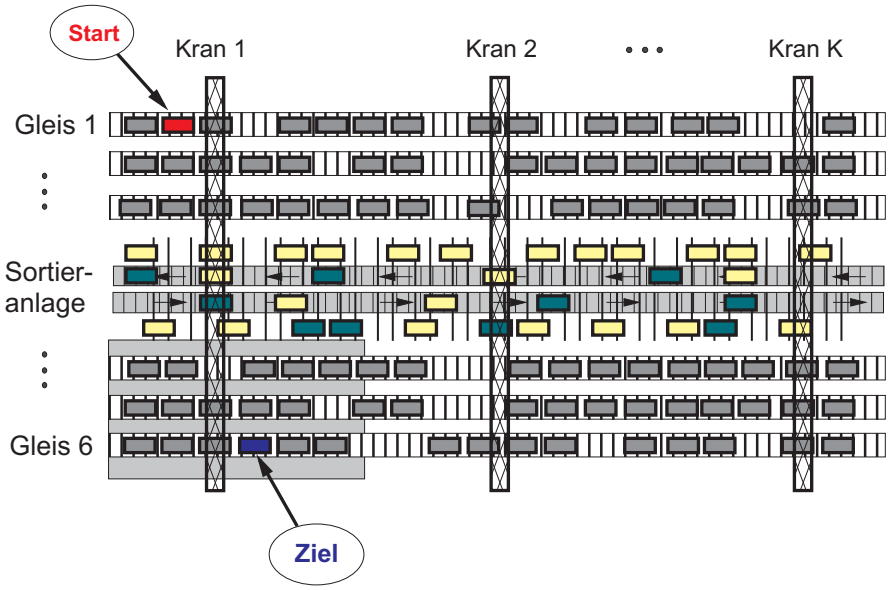


Abbildung 7.7.: Umschlag nach Kategorie 2.

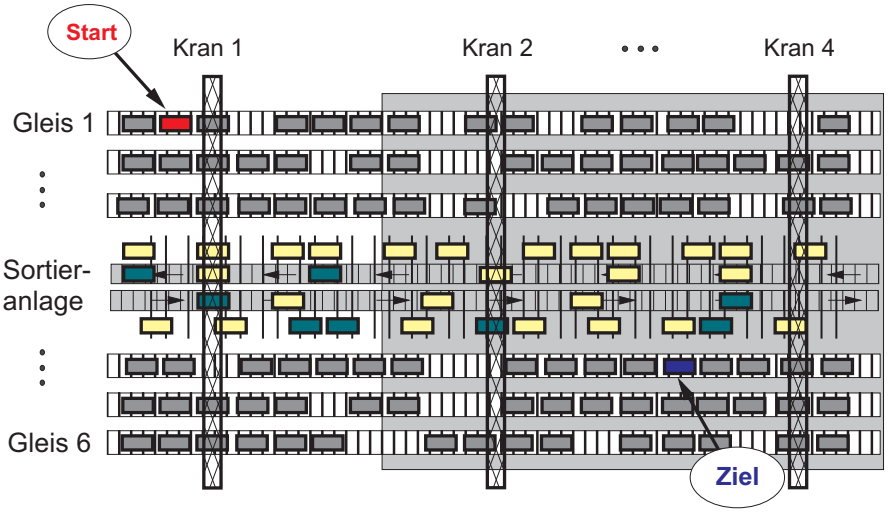


Abbildung 7.8.: Umschlag der Kategorie 3.

7.2.3. Direkter versus indirekter Umschlag

Alle Ladeeinheiten, die in Kategorie 2 fallen, sollten direkt umgeschlagen werden, wenn sich die Aufenthaltszeiten der Züge im Terminal überlappen. Die Entscheidung, ob direkt oder indirekt umgeschlagen wird, sollte während der Erzeugung der Reihenfolge erfolgen. Wird der direkte Umschlag mit einer und der indirekte mit zwei Operationen modelliert, ensteht das Problem der reihenfolgeabhängigen Anzahl von Operationen oder die Entscheidung, in welchem *Modus* der Auftrag auszuführen ist. Mit der Einführung eines Modus wird wiederum ein Zuordnungsproblem erzeugt, die Komplexität des Modells steigt.

Hier soll ein Weg verfolgt werden, der eine elegante Modellierung nutzt und gleichzeitig Rechenzeit spart. Zur Pufferung einer Ladeeinheit wird eine Position P_i auf dem Sortierspeicher ausgewählt, die auf der Strecke von der Start-Position P_s zu der Ziel-Position P_z der Ladeeinheit liegt. Da der Auftrag in Kategorie 2 fällt, werden zwei Operationen, O_1 und O_2 erzeugt. Jede Operation besteht aus den vier Teil-Vorgängen Leerfahrt, Aufnehmen, Vollfahrt und Absetzen. Werden die beiden Operationen *nicht* direkt nacheinander ausgeführt, ergibt sich die in Abbildung 7.9 oben dargestellte Situation. Die Gesamtzeit, die für den Umschlag benötigt wird, ist

$$t = t_1^{leer} + t_1^{auf} + t_1^{voll} + t_1^{ab} + t_2^{leer} + t_2^{auf} + t_2^{voll} + t_2^{ab} \tag{7.7}$$

Die Zeiten $t_1^{ab} + t_2^{leer} + t_2^{auf}$ können eingespart werden, wenn die Operationen O_1 und O_2 direkt hintereinander ausgeführt werden. Das ist möglich, wenn die Übergangszeit als Summe der Zeiten für die Vorgänge $t_1^{ab} + t_2^{leer} + t_2^{auf}$ und die Vollfahrt t_1^{voll}, t_2^{voll} als Bearbeitungszeit auf der Ressource modelliert wird. Wird die Übergangszeit 0, so entfallen das Absetzen, eine Leerfahrt und das Aufnehmen der Ladeeinheit. Die Vollfahrten t_1^{voll} und t_2^{voll} werden dann direkt hintereinander ausgeführt.

Neben der reihenfolgeabhängigen Übergangszeit zur Modellierung der Leerfahrt, die von der aktuellen Position des Kranes und der anzufahrenden Position abhängt, kommt die Übergangszeit zur Modellierung von direkten/indirekten Umschlägen hinzu (vgl. Kapitel 7.3).

7.2.4. Überlappende Einsatzbereiche - alternative Kräne

Den Kränen können *disjunkte* (harte Grenzen) oder *überlappende* (flexible Grenzen) Einsatzbereiche zugewiesen werden. Die Zuordnung, welcher Kran welche Ladeeinheit umschlägt, ist bei disjunkten Bereichen eindeutig, es kann jedoch zu ungünstigen Zuweisungen kommen, wie in Abbildung 7.10 dargestellt ist. Die

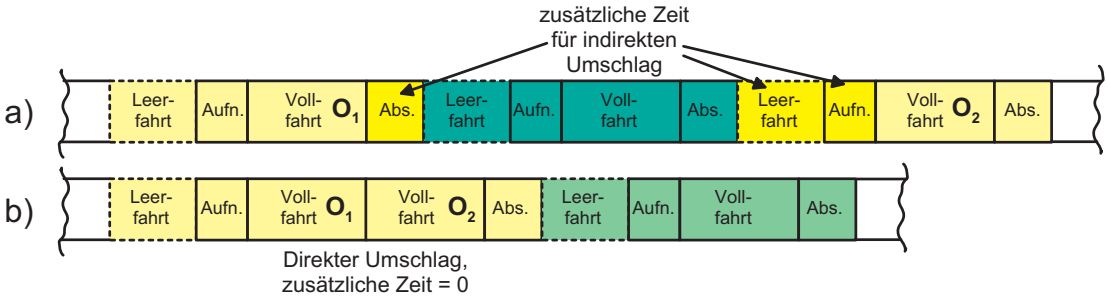


Abbildung 7.9.: Modellierung von indirektem (a) und direktem (b) Umschlag bei Klassifizierung als Kategorie zwei durch Verwenden von Operationen mit reihenfolgeabhängigen Übergangszeiten.

beiden dargestellten Aufträge entsprechen Kategorie 3, wenn S die Start-Position und Z_1, bzw. Z_2 die Ziel-Positionen Ziel_1, bzw. Ziel_2 bezeichnen. Wird mit $\xrightarrow{i}$ ein Umschlag dargestellt, der von Ressource i ausgeführt wird, sind folgende Operationen nötig:

$$\text{Ziel 1: } S \xrightarrow{i} FTF(S) \xrightarrow{FTF} FTF(Z_1) \xrightarrow{i+1} Z_1$$
$$\text{Ziel 2: } S \xrightarrow{i} FTF(S) \xrightarrow{FTF} FTF(Z_2) \xrightarrow{i+1} Z_2$$

Durch die disjunkten Bereiche ist die Zuweisung zu den Kränen i und $i+1$ eindeutig. Nun werden die Bereichsgrenzen flexibel gestaltet, der Parameter r stellt die Anzahl der Reihen der angrenzenden Bereiche dar, die als alternativ definiert werden. Im Beispiel können bei $r = 2$ vier Reihen Ladeeinheiten alternativ von Kran i oder $i+1$ umgeschlagen werden. Der Vorteil ist, daß Umschläge, die zuvor als Kategorie 3 klassifiziert wurden, nun Kategorie 2 oder 1 entsprechen, da sie von einem Kran ausgeführt werden können. Es ergeben sich die in Tabelle 7.1 aufgeführten Einsparungen. Index n bezeichnet den neuen erzeugten Umschlagsauftrag, der von Kran i ausgeführt wird.

$$\text{Ziel 1: } \Delta t = t_1 - t_n = t^{voll}_{1,i} + t^{voll}_{1,FTF} + t^{leer}_{1,i+1} + t^{auf}_{1,i+1} + t^{voll}_{1,i+1} + t^{ab}_{1,i+1} - t^{voll}_{n,i} \tag{7.8}$$

$$\text{Ziel 2: } \Delta t = t_2 - t_n = t^{voll}_{2,FTF} + t^{leer}_{2,i+1} + t^{auf}_{2,i+1} + t^{ab}_{2,i+1} \tag{7.9}$$

Die Zuweisung der Operationen zu den Ressourcen sollte abhängig von der Auslastung dynamisch erfolgen und wird daher in die Problem-Formulierung aufgenommen. Weiterhin ist durch Constraints sicherzustellen, daß es zu keiner Kollision in dem überlappenden Bereich kommt. In Abbildung 7.11 ist ein solcher Problemfall

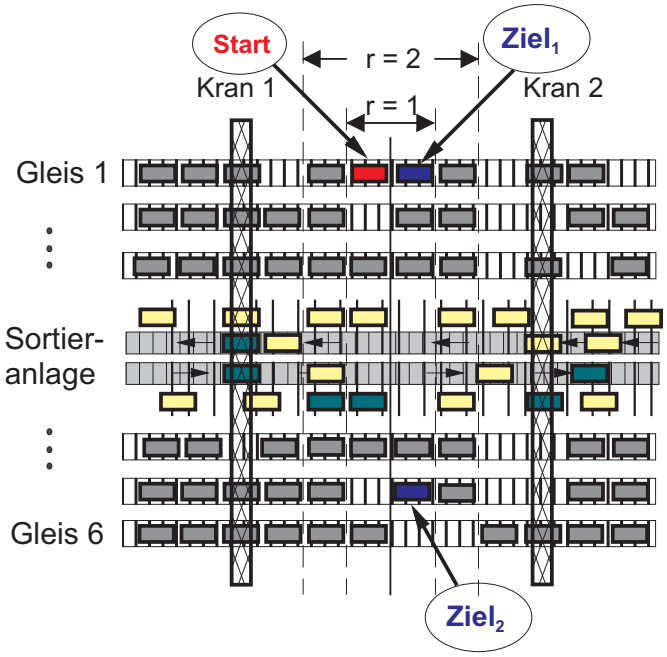


Abbildung 7.10.: Definition von flexiblen Bereichsgrenzen.

Kat.	Umschlag	Kran i	Sortiersp.	Kran $i+1$
1	$S \rightarrow Z_1$	$\Delta t = t_1^{voll} - t_{n,i}^{voll}$	$\Delta t = t_{1,FTF}^{voll}$	$\Delta t = t_{1,i+1}^{leer} + t_{1,i+1}^{auf} + t_{1,i+1}^{voll} + t_{1,i+1}^{ab}$
2	$S \rightarrow Z_2$	$\Delta t = 0$	$\Delta t = t_{2,FTF}^{voll}$	$\Delta t = t_{2,i+1}^{leer} + t_{2,i+1}^{auf} + t_{2,i+1}^{ab}$

Tabelle 7.1.: Einsparpotentiale, angegeben als Δt bei Verwendung von alternativen Bereichen.

anhand eines x-t-Diagrammes dargestellt. Bewegen sich die beiden angrenzenden Kräne zugleich in den Überlappungbereich, kommt es zu einer Kollision. Wird durch ein Constraint sichergestellt, daß nur der umschlagende Kran in dem Überlappungsbereich operiert, kann es bei großen Bereichsüberschneidungen zu ungewollten Stillstandzeiten kommen, da ein Kran nicht mehr in die angrenzenden Bereiche einfahren darf.

Hier bietet sich eine dynamische Sperrung an. Abhängig von der Position und der Fahrrichtung des Kranes werden Bereiche, die durch Reihen von Ladeeinheiten definiert sind, gesperrt bzw. entsperrt. Somit ist es möglich, daß zwei benachbarte Kräne zugleich in einem (ausreichend großen) alternativen Bereich operieren. Das Prinzip ist in Abbildung 7.12 dargestellt. Wird eine neue Reihe von einem Kran befahren, muß sie für den anderen Kran gesperrt werden. In Kapitel 7.3 werden die entsprechenden (dynamischen) Constraints formuliert.

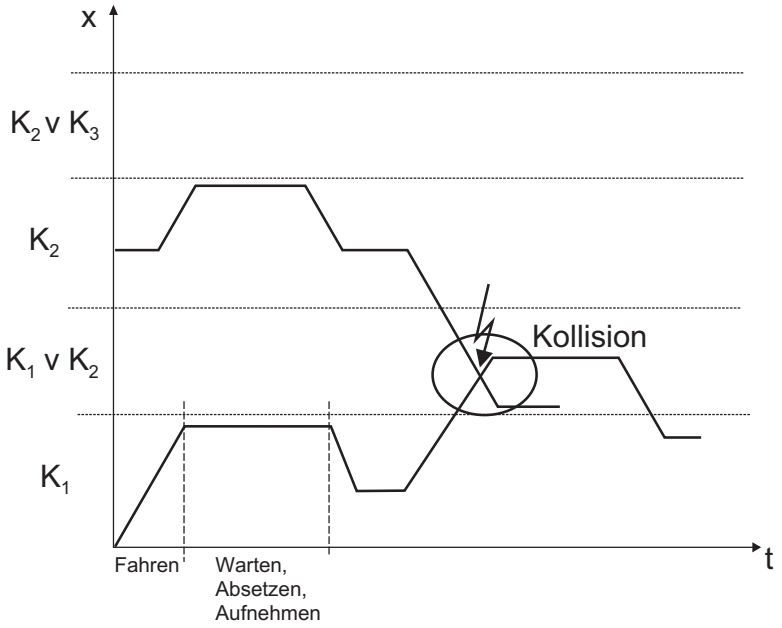


Abbildung 7.11.: Konflikte bei überlappenden Kranbereichen.

Sind die umzuschlagenden Ladeeinheiten nicht mehr den Kränen zugewiesen, so ist die Klassifizierung aus Kapitel 7.2.2 nicht mehr eindeutig. Die folgenden Überlegungen erweitern die Klassifizierung auf alternative Bereiche. Hierzu wird für jeden Kran i die Kernzone Z_i^k und für jedes Kranpaar die Zone der flexiblen Bereichsgrenzen Z_i^r festgelegt, wobei die angrenzenden Kräne $i-1$ und i sind. Die Zonen mit den zugehörigen Kranpaaren sind in Abbildung 7.13 dargestellt. Abhängig von der Start- und Zielposition einer Ladeeinheit o läßt sich die Zuweisung zu einem Kran oder einem Kranpaar bestimmen, siehe Tabelle 7.2. Diese dient als Grundlage für die anschließende Klassifizierung.

Start, $P_s(o)$	Z_i^k	Z_i^r	Z_{i+1}^r	Z_i^r
Ziel, $P_z(o)$	$Z_i^k \vee Z_{i-1}^r \vee Z_i^r$	$Z_i^k \vee Z_{i-1}^r$	$Z_i^k \vee Z_i^r$	$Z \backslash (Z_i^k \vee Z_{i+1}^k \vee Z_{i-1}^r)$
Zuw. zu Kran	i	i	i	$i \vee i+1$

Tabelle 7.2.: Zuweisung von umzuschlagenden Ladeeinheiten zu Kränen. Diese Zuweisung dient als Grundlage für die weitere Klassifizierung. Mit Z werden alle Zonen, also der gesamte Umschlagsbereich bezeichnet.

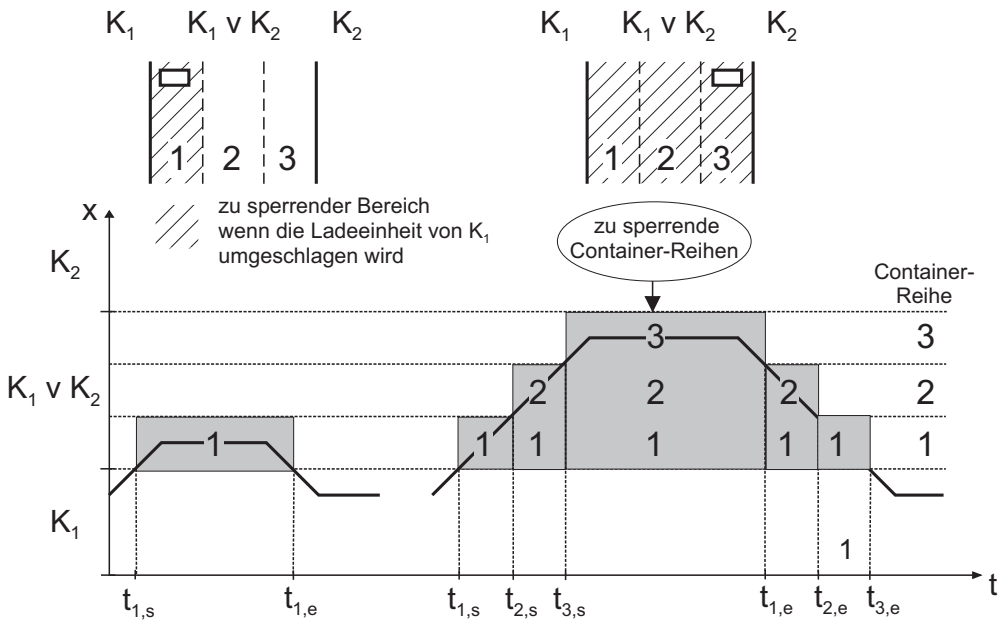


Abbildung 7.12.: Dynamisches Sperren bei überlappenden Kranbereichen.

Abschätzung der Einsparungen

Im folgenden soll abgeschätzt werden, wieviel Umschlagzeit sich durch flexible Bereichsgrenzen in Abhängigkeit von der Anzahl der eingesetzten Kräne einsparen läßt. Bei disjunkten Krangrenzen hat jeder Kran $C_k = \frac{C}{K}$ Ladeeinheiten umzuschlagen. Mit den flexiblen Bereichsgrenzen wird die Anzahl der *möglichen umzuschlagenden* Ladeeinheiten größer, mit r als Anzahl der alternativen Reihen ergeben sich $C_{k,alt} = \frac{C}{K} + 2r \cdot G$ umschlagbare Ladeeinheiten[7]. Die Wahrscheinlichkeit, daß ein Direktumschlag ausgeführt werden kann, steigt von $p_d = \frac{1}{K}$ zu: $p_d = \frac{1}{K} + \frac{2r \cdot G}{C}$. Die Wahrscheinlichkeit eines indirekten Umschlages ergibt sich somit zu $p_i = 1 - p_d$.

Der Erwartungswert des Umschlages $E(t_g)$ setzt sich zusammen aus den Erwartungswerten für Direktumschlag $E(t_{d,r})$, Indirektumschlag $E(t_i)$, das Aufnehmen/Absetzen $E(t_a)$ und Leerfahrten. In dem Überlappungsbereich werden zusätzliche Direktumschläge durchgeführt, somit wächst p_d in Abhängigkeit von r. Es werden keine zusätzlichen indirekten Umschläge (des benachbarten Kranes) ausgeführt, was bei der Berechnung des Erwartungswertes der Leerfahrt zu berücksichtigen ist. Die Leerfahrten werden aufgeteilt in solche, die abhängig von r, $E(t_{l,d}) = E(t_{d,r})$ (für die Direkt-Umschläge) und solche, die unabhängig von r,

[7] r ist zu verstehen pro Krangrenze, daher der Faktor 2 in der Formel.

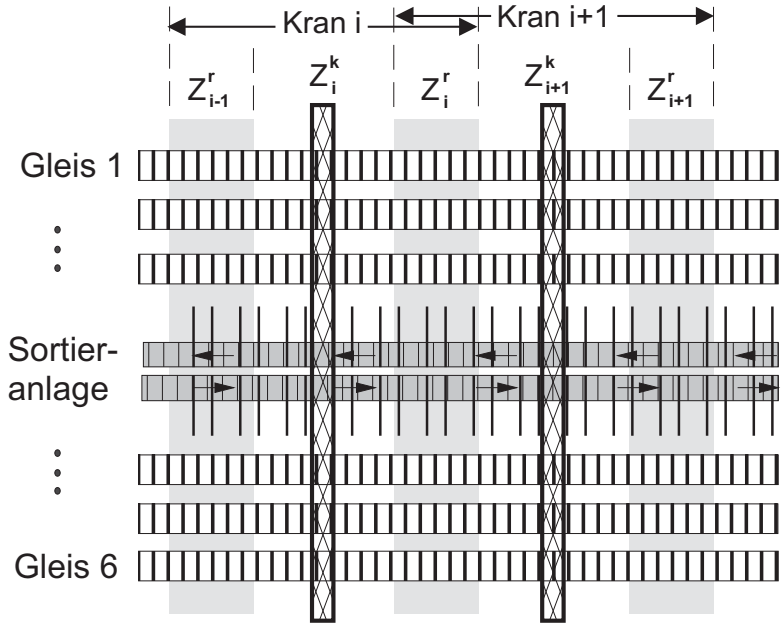


Abbildung 7.13.: Alternative Kranbereiche zur Zuweisung der Operationen.

$E(t_{l,i}) = E(t_{d,r=0})$ (für die indirekten Umschläge) sind.
Die Erwartungswerte lassen sich folgendermaßen bestimmen:

- $E(t_{d,r})$: Sei S die Start- und Z die Zielposition eines Umschlages. Es wird unterstellt, daß alle (S, Z) - Kombinationen eines Kranbereiches mit der gleichen Wahrscheinlichkeit $p(S, Z) = \frac{1}{C_k^2}$ angefahren werden, somit gilt

$$E(t_{d,r}) = p(S, Z) \cdot \sum_{S \in Z_i^k} \sum_{Z \in Z_i^k} t(S, Z) \tag{7.10}$$

- $E(t_i)$: Bei einem indirekten Umschlag ist nur die Katzbewegung des Kranes zeitbestimmend, also ist $E(t_i)$ unabhängig von r. Es wird davon ausgegangen, daß der nächste Pufferplatz auf dem Sortierspeicher frei ist. Somit ergibt sich der Erwartungswert aus den Fahrzeiten von den Gleisen zu dem nächstgelegenen Puffer des Sortierspeichers zu

$$E(t_i) = \frac{1}{G} \sum_{i=1}^{G} t(g_i, Sort) \tag{7.11}$$

- $E(t_a)$: Das Aufnehmen und Absetzen der Ladeeinheit ist unabhängig von r:

$$E(t_a) = t^{auf} + t^{ab} \tag{7.12}$$

Bei einem direkten Umschlag ist nur ein Vorgang, bei einem indirekten Umschlag sind zwei Vorgänge nötig, es ergibt sich für den Erwartungswert des Umschlages

$$E(t_g) = p_{d,r}(E(t_{d,r}) + E(t_a)) + 2p_{i,r}(E(t_i) + E(t_a)) + p_{d,r}E(t_{l,d}) + 2p_{i,r}E(t_{l,i}) \tag{7.13}$$

Abbildung 7.14 zeigt die Abhängigkeit der Wahrscheinlichkeit für einen Direktumschlag von der Anzahl der eingesetzten Kräne und der Größe des alternativen Bereiches (in Reihen von Ladeeinheiten). In Abbildung 7.15 ist $E(t_g)$ in Abhängig-

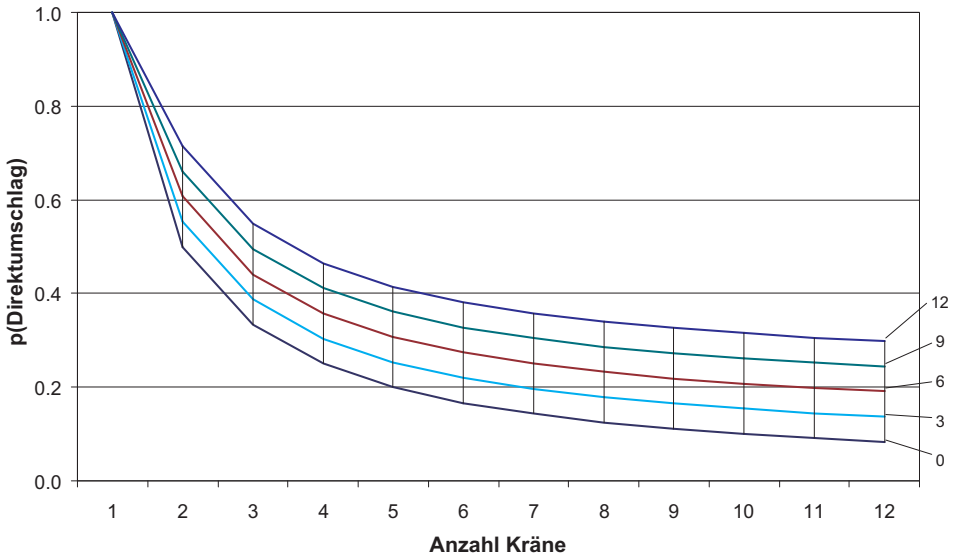


Abbildung 7.14.: Wahrscheinlichkeit eines Direktumschlages in Abhängigkeit der Anzahl eingesetzter Kräne und Größe des überlappenden Bereiches.

keit der eingesetzten Kräne und den alternativen Reihen aufgetragen. Deutlich ist ein Minimum für jeden Kran zu erkennen, durch das Layout des Terminals sind den überlappenden Kranbereichen jedoch Grenzen gesetzt. In Kapitel 7.5 werden diese Abschätzungen mit Zahlen untermauert, die Optimierung hat das Ziel, die Leerfahrten weiter zu reduzieren.

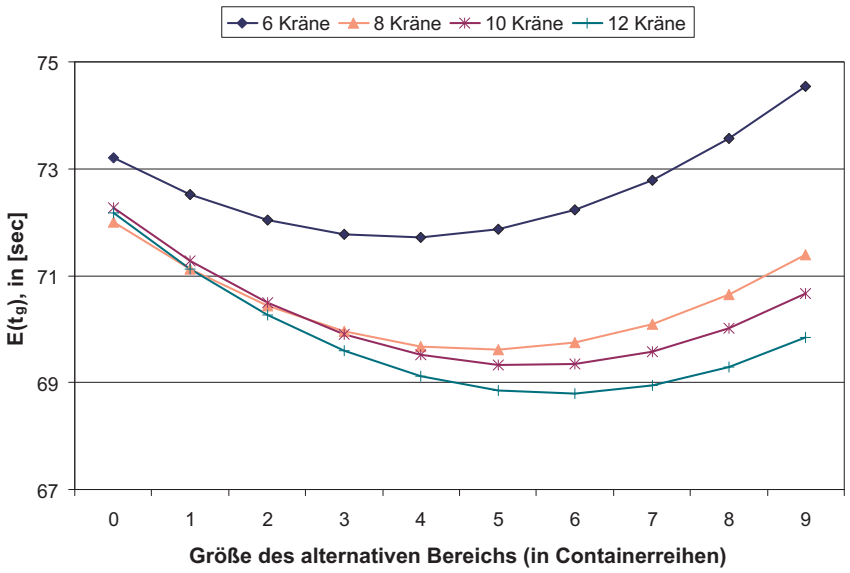


Abbildung 7.15.: Abschätzung der Einsparungen bei Verwendung von alternativen Kränen.

7.3. Formulierung als Constraint Optimization Problem

Die Elemente des Systems sind die Operationen und die zur Ausführung zur Verfügung stehenden Ressourcen. Ein (Umschlags-)Aufträge setzt sich aus Operationen zusammen. Das Ziel der Optimierung ist die Bestimmung der Umschlagsreihenfolge der Ladeeinheiten, somit die Bestimmung der Startzeitpunkte der Operationen. Constraints werden durch die von den Aufträgen und der Belegung gegebenen Reihenfolge-Beziehungen, durch die Kapazitätsbeschränkung der Ressourcen und die alternative Zuordnung von Operationen zu Ressourcen gegeben. Im folgenden wird das beschriebene System mit den gegebenen Randbedingungen als Constraint Optimization Problem formuliert.

Die Objektstruktur ist in diesem Anwendungsfall deutlich einfacher als in Kapitel 6, dafür sind die Kapazitäts-Constraints wesentlich komplexer. Auch ist keine eindeutige Materialflußrichtung vorgegeben, die Ressourcen interagieren permanent.

Die Definition des mehrstufigen Umschlagproblems kann auf das vorliegende Terminal angewendet werden. Die (Umschlags-)Aufträge $j \in \mathcal{J}$ entsprechen den umzuschlagenden Ladeeinheiten. Die Operationen $O \in \mathcal{O}$ sind die nach der Klassifizierung (Kapitel 7.2.2) aufgeteilten Aufträge. Die Kräne werden aus Gründen

der Übersichtlichkeit mit $k_1, \ldots, k_K \in \mathcal{K}$ bezeichnet und sind Ressourcen mit der Kapazität $cp(k) = 1, \ \forall \ k \in \mathcal{K}$, wobei $\mathcal{K} \subset \mathcal{R}$. Die alternativen Ressourcen werden aus je zwei benachbarten Kränen gebildet und mit $k_1^{alt}, \ldots, k_{K-1}^{alt} \in \mathcal{K}^{alt}$ bezeichnet. Es ist einleuchtend, daß bei K Kränen $K-1$ alternative Mengen entstehen. Jedes Fahrzeug der Sortieranlage stellt eine Kombination aus einer puffernden und bearbeitenden Ressource dar. Die FTF werden zusammengefaßt zu einer Ressource $r_s \in \mathcal{R}$, deren Kapazität der Anzahl an Fahrzeugen f entspricht, $cp(r_s) = f$. Der Fahrvorgang der FTF und damit die evtl. auftretenden Staueffekte werden hier nicht beachtet. In der realen Anlage ist der Unterschied der Investitionsausgaben Kran zu FTF so hoch, daß es sich nicht lohnen würde, die Sortieranlage als Engpaß des Systems vorzusehen. Aufgrund der Kombination aus bearbeitender und puffernder Ressource und der begrenzten Kapazität muß der Sortierspeicher als Kapazität berücksichtigt werden.

Die Ladeeinheiten werden von einem Start-Zug auf einen Ziel-Zug umgeschlagen, die Umschlagsmatrix und damit die Positionen auf den Zügen sind bekannt. Die Menge der Züge wird mit $z \in \mathcal{Z}$ bezeichnet, die Zuordnung eines Auftrages zu einem Zug geschieht über die Abbildung $a : \mathcal{J} \rightarrow \mathcal{Z}$. Zur Vereinfachung wird die Abbildung erweitert auf $a_s : \mathcal{J} \rightarrow \mathcal{Z}$ für den Start-Zug der entsprechenden Ladeeinheit und $a_z : \mathcal{J} \rightarrow \mathcal{Z}$ für den Ziel-Zug[8]. Der Fahrplan, nach dem die Züge operieren, wird durch $t^{rel} : \mathcal{Z} \rightarrow \mathbb{N}^+$ für die Einfahrzeit und $t^{due} : \mathcal{Z} \rightarrow \mathbb{N}^+$ für die Ausfahrzeit bestimmt.

Das Ziel der Optimierung ist die Bestimmung der Startzeitpunkte der Operationen. Die Constraint-Variablen sind die Entscheidungsvariablen des Optimierungsproblems und entsprechen somit den Startzeitpunkten t_s der einzelnen Operationen. Die Wertebereiche der Variablen sind initial $D_{t_s} = [0, \ldots, H]$ und werden, wie im folgenden Kapitel dargestellt, durch t^{rel} und t^{due} bestimmt. Die Constraints werden mit $c_1, \ldots, c_n$ bezeichnet.

7.3.1. Temporale Constraints durch den Fahrplan

Die Operationen werden aus den Aufträgen entsprechend der Klassifizierung erzeugt. Somit lassen sich unter Verwendung des Fahrplanes temporale (einwertige) Constraints angeben, die t^{rel} und t^{due} der Operationen und somit die Wertebereiche bestimmen.

[8] Wobei die Ladeeinheit auf einem Tragwagen des Start-Zuges in das Terminal einfährt und es auf einem Tragwagen des Ziel-Zuges verläßt. Jeder Zug kann hierbei die Funktion des Start- und Zielzuges haben, daher wird die Zuordnung über den Auftrag getroffen.

- Kategorie 1: Es wird eine Operation O_1^j (Abladen/Aufladen) erzeugt.

$$t^{rel}(O_1^j) = \text{MAX}\,[t^{rel}(a_s(j)); t^{rel}(a_z(j))] \quad (7.14)$$
$$t^{due}(O_1^j) = \text{MIN}\,[t^{due}(a_s(j)); t^{due}(a_z(j))] \quad (7.15)$$

- Kategorie 2: Zwei Operationen O_1^j (Abladen), O_2^j (Aufladen) werden erzeugt.

$$t^{rel}(O_1^j) = t^{rel}(a_s(j)) \quad (7.16)$$
$$t^{due}(O_1^j) = \text{MIN}\,[t^{due}(a_s(j)); t^{due}(a_z(j)) - pt(O_2^j)] \quad (7.17)$$
$$t^{rel}(O_2^j) = t^{rel}(a_z(j)) - pt(O_2^j) \quad (7.18)$$
$$t^{due}(O_2^j) = t^{due}(a_z(j)) \quad (7.19)$$

- Kategorie 3: Es werden drei Operationen O_1^j (Abladen), O_2^j (Sortierspeicher) und O_3^j (Aufladen) erzeugt.

$$t^{rel}(O_1^j) = t^{rel}(a_s(j)) \quad (7.20)$$
$$t^{due}(O_1^j) = \text{MIN}\,[t^{due}(a_s(j)); t^{due}(a_z(j)) - (pt(O_2^j) + pt(O_3^j)] \quad (7.21)$$
$$t^{rel}(O_2^j) = t^{rel}(a_s(j)) + pt(O_1^j) \quad (7.22)$$
$$t^{due}(O_2^j) = t^{due}(a_z(j)) - pt(O_3^j) \quad (7.23)$$
$$t^{rel}(O_3^j) = \text{MAX}\,[t^{rel}(a_z(j)); t^{rel}(a_s(j))] \quad (7.24)$$
$$t^{due}(O_3^j) = t^{due}(a_z(j)) \quad (7.25)$$

Es gilt für die Start- und Endzeitpunkte jeder Operation:

$$t_s(O_i^j) \geq t^{rel}(O_i^j) \quad (7.26)$$
$$t_e(O_i^j) \geq t^{due}(O_i^j) \quad (7.27)$$

7.3.2. Reihenfolge-Constraints

Neben den temporalen lassen sich aufgrund der Klassifizierung die folgenden Reihenfolge-Constraints angeben:

- Kategorie 2: $c_r : O_1^j \succ O_2^j$
- Kategorie 3: $c_r : O_1^j \succ O_2^j \succ O_3^j$

7.3.3. Constraints durch Belegung des Zielplatzes

Soll eine Ladeeinheit umgeschlagen werden, muß die Ziel-Position leer sein, ansonsten kommt es zu einer Kollision. Das kann durch ein Reihenfolge-Constraint verhindert werden, das die Start- und Zielposition einer Ladeeinheit nutzt. Das folgende Reihenfolge-Constraint stellt sicher, daß zuerst die Ladeeinheit der Zielposition umgeschlagen wird:

$$c_s : O_i^j \succ O_l^k \quad \forall j,i,k,l \mid P_z(O_i^j) = P_s(O_l^k) \tag{7.28}$$

7.3.4. Constraints zur Modellierung von Direkt- und Indirektumschlag

In Kapitel 7.2.3 wurde die Modellierung vorgestellt, um Direkt- und Indirektumschläge durch Nutzen von reihenfolgeabhängigen Übergangszeiten t^{set} zu modellieren. Werden von einem Kran zwei Operationen eines Auftrages unmittelbar hintereinander ausgeführt, liegt ein Direktumschlag vor, und die Übergangszeit ist 0. Andernfalls ergibt sich die Übergangszeit aus der Zeit für das Absetzen der Ladeeinheit, der Leerfahrt zur nächsten Position und dem Aufnehmen der nächsten Ladeeinheit. Die Übergangszeit zwischen den Operationen O_i^j und O_l^k, $t^{set}(O_i^j, O_l^k)$ wird definiert als

$$t^{set}(O_i^j, O_l^k) = \begin{cases} 0 & \text{für } O_i^j \gg O_l^k \wedge j = k \\ t(P_z(O_i^j), P_s(O_l^k)) + t^{auf} + t^{ab} & \text{sonst} \end{cases}$$
$$\forall \quad ra(O_i^j) = ra(O_l^k) \tag{7.29}$$

Die Relation $\gg$ soll darstellen, daß die beiden Operationen *direkt* nacheinander ausgeführt werden, somit gilt $t_s(O_l^k) = t_e(O_i^j)$. Dies entspricht der Intervall-Relation *meets* aus Abbildung 4.2. Eine allgemeine Reihenfolge im Schedule wird durch $\succ$ ausgedrückt, hier gilt $t_s(O_l^k) > t_e(O_i^j)$. Diese Relation entspricht *before* aus Abbildung 4.2.

In Kapitel 7.2.1 wurde die Zeit zur Vorbereitung des Spreaders auf die aufzunehmende Ladeinheit dargestellt, die ebenfalls reihenfolgeabhängig ist. Die benötigte Zeit t^v muß beachtet werden, falls sie größer als die Fahrzeit ist. Es gilt:

$$t(P_z(O_i^j), P_s(O_l^k)) = \begin{cases} t^v(O_i^j, O_l^k) & \text{für } \alpha \\ t^{fahr}(P_z(O_i^j), P_s(O_l^k)) & \text{für } \beta \end{cases} \tag{7.30}$$
$$\alpha \Leftrightarrow t^{fahr}(P_z(O_i^j), P_s(O_l^k)) < t^v(O_i^j, O_l^k)$$
$$\beta \Leftrightarrow t^{fahr}(P_z(O_i^j), P_s(O_l^k)) \geq t^v(O_i^j, O_l^k)$$

7.3.5. Kapazitäts-Constraints

Die in Kapitel 4 und 5.1 beschriebenen Kapazitäts-Constraints müssen erfüllt sein. Für die Kräne, als Ressource mit $cp(r) = 1$, muß das Constraint (4.16) erfüllt sein, d.h. zwei Operationen dürfen sich nicht überschneiden. Übergangszeiten werden hierbei berücksichtigt.

Constraints zur Modellierung des Sortierspeichers

Der Sortierspeicher hat eine Kapazität von $cp(r_s) = f$. Die Leerfahrt des FTF zu der Aufnahme-Position wird nicht in die Modellierung mit aufgenommen, wie bereits angesprochen wurde. Die Kapazität des Sortierspeichers wird zu dem Zeitpunkt belegt, ab dem der Kran die Zielposition (d.h. über dem FTF) erreicht hat. Die Bearbeitungszeit des FTF ist die Fahrzeit zu der Zielposition. Die Kapazität des Sortierspeichers wird zu dem Zeitpunkt wieder freigegeben, zu dem die Ladeeinheit von dem Kran aufgenommen wird. Die FTF stellen also eine Kombination aus bearbeitenden und puffernden Ressourcen dar, somit muß (4.18) erfüllt sein. Sei O_1^j die Operation des aufnehmenden Kranes (Start-Position), O_2^j die Operation des FTF des Sortierspeichers und O_3^j die des abliefernden Kranes (Ziel-Position). Die Bearbeitungszeit des Kranes $pt(O_1^j)$, bzw. $pt(O_3^j)$ entsprechen der Vollfahrt. Leerfahrt, Aufnehmen und Absetzen sind die Übergangszeiten. Somit stellt $t_e(O_1^j)$ die Zeit dar, zu der der Kran über dem Fahrzeug angekommen ist, das FTF des Sortierspeichers wird zu der richtigen Zeit belegt. Das FTF muß auf der Zielposition bereitstehen, sobald der Kran die Position des FTF erreicht hat. Die Kapazität steht nach Aufnehmen der Ladeeinheit, also bei Start der Vollfahrtzeit $t_s(O_3^j)$ wieder zur Verfügung. Das Constraint (4.18) muß somit angepaßt werden zu:

$$
\begin{aligned}
c_{k_1} &: t_e(O_1^j) + pt(O_2^j) \leq t_s(O_3^j) - t^{ab} \\
c_{k_2} &: \sum_{O_2^j \mid (O_1^j \succ O_2^j \succ O_3^j) \wedge (t_e(O_1^j) < t < t_s(O_3^j))} sz(O_2^j) \leq cp(r_s) \quad \forall\, j \in F_3,\ 0 \leq t \leq H
\end{aligned}
\tag{7.31}
$$

Hier gilt $sz(O_2^j) = 1$. Das Constraint muß nicht nur für Umschläge der Kategorie 3, sondern auch für Kategorie 2 aufgestellt werden, da der Sortierspeicher, d.h. ein FTF, das nicht fährt, als Puffer verwendet wird. Es muß für die Operationen $O_1^j \succ O_2^j$ gelten:

$$
c_{k_3} : \sum_{(O_1^l \succ O_2^l) \wedge (t_e(O_1^l) < t < t_s(O_2^l))} sz(O_1^l) \leq cp(r_s) \quad \forall\, l \in F_2,\ 0 \leq t \leq H \tag{7.32}
$$

Auch hier gilt $sz(O_1^l) = sz(O_2^l) = 1$. Durch Gleichung (7.29) ist die Beachtung der Übergangszeiten bei Direkt/Indirektumschlag bereits sichergestellt. Da sich beide Constraints (7.32) und (7.31) auf den Sortierspeicher beziehen, müssen sie zusammengefaßt werden, es ergibt sich:

$$c_{k_4}: \sum_{O_2^j \mid (O_1^j \succ O_2^j \succ O_3^j) \wedge (t_e(O_1^j) < t < t_s(O_3^j))} sz(O_2^j) + \sum_{(O_1^l \succ O_2^l) \wedge (t_e(O_1^l) < t < t_s(O_2^l))} sz(O_1^l) \leq cp(r_s) \quad \forall\, j \in F_3 \ \wedge\ l \in F_2,\ 0 \leq t \leq H \tag{7.33}$$

Constraints zur Modellierung der alternativen Ressourcen

Werden die Operationen alternativen Ressourcen zugewiesen, muß sichergestellt werden, daß sich die Kräne nicht gegenseitig während des Umschlages behindern. Eine solche Kollisionsgefahr würde zu Ausweichvorgängen und damit zu unnötigem Zeitverlust führen. Die Kollisionsfreiheit kann einfach sichergestellt werden, wenn Gleichung (4.9), also ein disjunktes Constraint, zwischen allen Ladeeinheiten des alternativen Bereiches erfüllt sein muß. Somit wird während des Umschlages einer Ladeeinheit des Überlappungsbereiches Z_i^k keine andere Ladeeinheit dieses Bereiches umgeschlagen. Formal gilt

$$\begin{aligned} &c_{a_1}: (o \succ p) \vee (p \succ o) \\ &\forall\, o, p, i \mid (P_s(o) \in Z_i^r \vee P_z(o) \in Z_i^r) \wedge (P_s(p) \in Z_i^r \vee P_z(p) \in Z_i^r) \end{aligned} \tag{7.34}$$

Wenn ein Kran keinen Kernbereich mehr hat, also $Z_i^k = \emptyset$ ist[9], ist durch das Constraint nur ein sehr eingeschränktes Arbeiten möglich. Die Hälfte der Kräne kann nicht arbeiten, da der Umschlag durch das Constraint (7.34) gesperrt ist.

In Abbildung 7.12 ist dargestellt, wie die Bereiche dynamisch gesperrt und wieder freigegeben werden können, abhängig von der Position der umzuschlagenden Einheit und der Kinematik des umschlagenden Kranes.

Constraints zur dynamischen Sperrung der Überlappungsbereiche

Bei einer großen Überlappung der Kranbereiche sollten die alternativen Reihen dynamisch gesperrt werden, wie in Abbildung 7.12 dargestellt wurde. Abhängig von dem anzufahrenden Ziel der Operation $P_z(O_i^j)$ werden die Container-Reihen

[9] z.B. da die beiden angrenzenden alternativen Zonen den gesamten Bereich abdecken.

gesperrt. Die Zielposition wird zu dem Zeitpunkt $t_e(O_i^j)$ erreicht, die komplette Container-Reihe der Zielposition wird ab dem Zeitpunkt $t_{z,s}(O_i^j) = t_e(O_i^j) - t^{fahr}(l_{ct})$, d.h. bei Eintritt des Kranes, gesperrt. l_{ct} ist die Länge eines Containers. Der Austritt aus der Zielreihe, d.h. das Entsperren, kann ab dem Zeitpunkt $t_{z,e}(O_i^j) = t_{z,s}(O_i^j) + t^{ab} + t^{auf} + t^{fahr}(l_{ct})$ erfolgen. Die Zeiten für die Reihen (in Fahrtrichtung des Kranes) vor der Zielreihe können nun ebenfalls angegeben werden zu

$$\begin{aligned} t_{z-1,s}(O_i^j) &= t_{z,s}(O_i^j) - t^{fahr}(l_{ct}) \\ t_{z-1,e}(O_i^j) &= t_{z,e}(O_i^j) + t^{fahr}(l_{ct}) \\ &\vdots \quad\quad \vdots \\ t_{z-n,s}(O_i^j) &= t_{z,s}(O_i^j) - t^{fahr}(n \cdot l_{ct}) \\ t_{z-n,e}(O_i^j) &= t_{z,e}(O_i^j) + t^{fahr}(n \cdot l_{ct}) \end{aligned}$$

Aufbauend auf diesen Zeiten kann angegeben werden, wann die Reihen durch Einführen eines Reihenfolge-Constraints gesperrt und freigegeben werden müssen. Alle Container, deren Start- und Zielposition in den alternativen Reihen liegen, dürfen in den angegebenen Zeiträumen nicht umgeschlagen werden, Constraint (7.34) kann erweitert werden. Zur Definition des Constraints wird die Reihe der Position einer Operation $r_z(o) \in [1, \ldots, r]$, immer in Abhängigkeit von dem umschlagenden Kran, eingeführt. Diese Reihe kann die Start- oder die Zielposition eines Umschlages sein. Somit lassen sich die alternativen Reihen $1, 2, \ldots, r_z(o)$ mit Hilfe der folgenden Reihenfolge-Constraints sperren.

$$\begin{aligned} &c_{a2} : t_s(p) > t_{z-\Delta r,e}(o) \ \vee \ t_e(p) < t_{z-\Delta r,s}(o) \\ &\forall\, o, p \mid r_z(o) > r_z(p), \text{mit } \Delta r = r_z(o) - r_z(p) \end{aligned} \tag{7.35}$$

7.3.6. Einfache Zuordnungs-Heuristik für alternative Ressourcen

Um die detaillierten Constraints zur Sicherung der Kollisionsfreiheit der Überlappungsbereiche angeben zu können, muß die Zuweisung der Operationen zu den Kränen feststehen. Mit der Verwendung dynamischer Constraints, die während des Suchprozesses hinzugefügt oder entfernt werden, kann die Zuweisung auch während des Suchprozesses stattfinden.

Im folgenden wird eine einfache Heuristik vorgestellt, die den Kränen die Operationen zuweist. Das Ziel ist eine ausgeglichene Belastung der Kräne. Nach Tabelle

7.2 können die Operationen in eindeutige oder alternative Zuweisung unterschieden werden. Die Belastung V_i eines Kranes i durch die eindeutig zugewiesenen Operationen $(ra(o) = i)$ läßt sich bestimmen zu:

$$V_i = \sum_{o|ra(o)=i} pt(o) + (t^{auf} + t^{ab})_{o \in F_1 \cup F_3} + \frac{1}{2}(t^{auf} + t^{ab})_{o \in F_2} \tag{7.36}$$

Die Übergangszeiten sind reihenfolgeabhängig, daher werden nur die Zeiten für das Aufnehmen und Absetzen berücksichtigt. Bei Operationen der Kategorie 2, also $o \in F_2$ wird $\frac{1}{2}(t^{auf} + t^{ab})$ als untere Schranke angegeben, da die Entscheidung direkter oder indirekter Umschlag ebenfalls abhängig von der Reihenfolge ist. Die Operationen, deren Start- oder/und Ziel-Position nahe an einem Kran liegen, sollten auch diesem zugewiesen werden, mit $\Delta x_{o,i}$ wird die Entfernung der Start- und Zielposition zu der Mitte des Kranbereiches angegeben:

$$\Delta x_{o,i} = |x_{s,o} - x_{m,i}| + |x_{z,o} - x_{m,i}| \tag{7.37}$$

In Algorithmus 2 wird die Zuweisungsheuristik angegeben. Die Menge $\mathcal{UO}_i$ definiert die Operationen, die noch keinem Kran eindeutig zugewiesen worden sind und in deren möglichen Zuweisungen Kran i enthalten ist, also $fr(o) = i$.

Algorithmus 2 Zuweisung der Operationen zu alternativen Kränen.

1: **for** $i = 1$ to $n - 1$ **do**
2: Bestimme V_i
3: **end for**
4: $\mathcal{UO} = \bigcup_{i=1}^{n-1} \mathcal{UO}_i$
5: **while** $\mathcal{UO} \neq \emptyset$ **do**
6: Wähle Kran i mit MIN $V_i \wedge \mathcal{UO}_i \neq \emptyset$
7: Wähle Operation $o \in \mathcal{UO}_i$ mit MIN $\Delta x_{o,i}$
8: $ra(o) = i$
9: $\mathcal{UO}_i \backslash \{o\}$
10: Update V_i
11: **end while**

7.3.7. Zielfunktion

Auch in dieser Anwendung wird, wie bereits in Kapitel 6.4.1, die Minimierung der maximalen Verspätung als Zielfunktion verwendet, um das COP zu formulieren.

Die Gleichung (7.27) wird, aufgeteilt nach den Kategorien, relaxiert zu:

$$t_e(O_1^j) = t^{due}(O_1^j) + l^j \text{ - Kategorie 1} \quad (7.38)$$
$$t_e(O_2^j) = t^{due}(O_2^j) + l^j \text{ - Kategorie 2} \quad (7.39)$$
$$t_e(O_3^j) = t^{due}(O_3^j) + l^j \text{ - Kategorie 3} \quad (7.40)$$

Analog zu den Ausführungen in Kapitel 6.4.1 wird die Formulierung des COP durch Hinzufügen des folgenden (Zielfunktions-)Constraints

$$c_z : l^j \leq L_{max} \; \forall \; t \quad (7.41)$$

vervollständigt. In Kapitel 7.5 wird das COP verwendet, um ein reales Terminal zu optimieren.

7.4. Untersuchungen zur Gleisbelegung

Ein wichtiger Gegenstand der Untersuchung ist die Belegung der Gleise: Bei gegebenem Fahrplan (Ankunfts-, und Abfahrtszeiten der Züge) ist zu entscheiden, auf welches Gleis die Züge einfahren. Bei der Gleisbelegung des Terminals sollten Züge, die sich in der Aufenthaltszeit überlappen und für die eine maximale Anzahl an Direktumschlägen möglich ist, benachbarten Gleisen zugewiesen werden. Die Anzahl könnte bei der Optimierung als Zielkriterium gelten. Durch die flexiblen Bereichsgrenzen ist die Bestimmung der Anzahl Direktumschläge abhängig von der tatsächlichen Umschlagsreihenfolge. Die Gleisbelegung beeinflußt wiederum die Bildung der Umschlagsreihenfolge. Es entsteht also ein Zyklus, der gelöst werden muß. Es wird eine *Umschlagskennziffer* definiert, die die *Umschlag-Beziehung* zweier Züge darstellt. Direktumschläge sind möglich, wenn die horizontale (x-) Entfernung der Start- zu der Zielposition der umzuschlagenden Ladeeinheit klein ist (siehe Abb. 7.16). Die exakte Grenze, ab der ein Direktumschlag möglich ist, kann nicht angegeben werden, da bei der Reihenfolgebildung mit den flexiblen Bereichsgrenzen gerechnet wird. Die Start-Position einer Ladeeinheit c wird mit $x_{s,c}$, die Ziel-Position mit $x_{z,c}$ bezeichnet. Bei dem Umschlag wird die x-Strecke $\Delta x_c = |x_{s,c} - x_{z,c}|$ zurückgelegt.

Es läßt sich eine Proportionalität zwischen der horizontalen Entfernung und der Möglichkeit zum Direktumschlag zeigen, die in der Umschlagskennziffer dargestellt ist. In der Zielfunktion wird die Umschlagskennziffer zusammen mit der vertikalen (y-) Entfernung benutzt, um die Züge den Gleisen zuzuordnen.

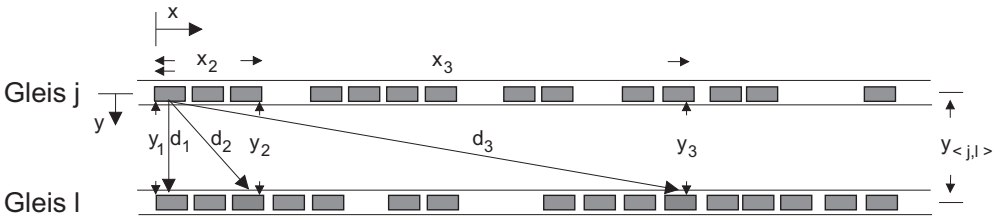


Abbildung 7.16.: Zur Erläuterung der Proportionalität zwischen (x-) Entfernung und der Anzahl möglicher Direktumschläge.

7.4.1. Definition der Umschlagskennziffer

Werden nur Umschläge mit $\Delta x_c = 0$ ausgeführt, sollten die Züge nebeneinander stehen. Wird die Umschlagskennziffer proportional zur Summe der Δx_c definiert, so bedeutet eine größere Umschlagskennziffer weniger Direktumschläge. Durch die Gewichtung mit der Anzahl der Ladeeinheiten und dem Abstand der Gleise werden bei einer Optimierung die größeren Umschlagskennziffern systematisch bevorteilt, da eine größere Verbesserung erzielt werden kann. Aber die größere Verbesserung sollte bei Zügen mit einer sehr großen Anzahl an möglichen Direktumschlägen geschehen, also wird die Umschlagskennziffer $U_{i,k}$ zwischen den Zügen i und k (kein gerichteter Umschlag, beide Richtungen sind möglich und berücksichtigt) definiert zu:

$$U_{i,k} = \frac{B}{n_{i,k} \cdot \sum_{c \in C_{i,k}} |x_{s,c} - x_{z,c}|} \tag{7.42}$$

B ist ein konstanter Faktor, um eventuell sehr kleine Werte auszugleichen, und damit numerische Probleme bei der Lösungsfindung zu vermeiden. $C_{i,k}$ ist die Menge der Ladeeinheiten, die zwischen Zug i und Zug k ausgetauscht werden, c der zugehörige Index. Mit $n_{i,k}$ wird die Anzahl Ladeeinheiten bezeichnet, die zwischen den beiden Zügen umgeschlagen werden, $d_{j,l}$ beschreibt den Abstand der beiden Gleise. Das Modell zur Maximierung der Anzahl Direktumschläge läßt sich formulieren als:

$$\text{MIN} \ : \sum_{i} \sum_{j} \sum_{k,k \neq i} \sum_{l,l \neq j} U_{i,k} \cdot d_{j,l} \cdot y_{ij} \cdot y_{kl} \tag{7.43}$$

Unter den Nebenbedingungen

$$y_{ij}(y_{kl}) = \begin{cases} 1 & \text{Zug } i(k) \text{ fährt auf Gleis } j(l) \text{ ein} \\ 0 & \text{sonst} \end{cases} \tag{7.44}$$

$$\sum_j y_{ij} = 1 \; \forall \; i \tag{7.45}$$

$$\sum_l y_{kl} = 1 \; \forall \; k \tag{7.46}$$

$$y_{ij}, y_{kl} \in \{0, 1\} \tag{7.47}$$

7.4.2. Integration der Überlappung der Aufenthaltszeiten der Züge

Bei der bisherigen Überlegung wurde die Überlappung der Aufenthaltszeiten der Züge noch nicht berücksichtigt. Wenn zwei Züge keinen überlappenden Aufenthalt im Bahnhof haben, ist es auch bei einer sehr starken Austauschbeziehung nicht sinnvoll, diese beiden Züge in benachbarte Gleise einfahren zu lassen.

Die Überlappung der Aufenthaltszeiten kann unterschiedlich sein. Im folgenden stellen $t^{rel}(i), t^{rel}(k)$ die Einfahrzeit und $t^{due}(i), t^{due}(k)$ die Ausfahrzeiten von Zug i, bzw. k dar. Somit ergeben sich folgende Zusammenhänge für die Überlappung der Aufenthaltszeiten $\Delta t_{i,k}$:

$$\begin{aligned}
\Delta t_{i,k} = 0 \quad &\forall \quad t^{rel}(i) > t^{due}(k) \\
\Delta t_{i,k} = 0 \quad &\forall \quad t^{rel}(k) > t^{due}(i) \\
\Delta t_{i,k} = t^{due}(i) - t^{rel}(k) \quad &\forall \quad t^{rel}(i) < t^{rel}(k) < t^{due}(i) < t^{due}(k) \\
\Delta t_{i,k} = t^{due}(k) - t^{rel}(i) \quad &\forall \quad t^{rel}(k) < t^{rel}(i) < t^{due}(k) < t^{due}(i) \\
\Delta t_{i,k} = t^{due}(i) - t^{rel}(i) \quad &\forall \quad t^{rel}(k) < t^{rel}(i) < t^{due}(i) < t^{due}(k) \\
\Delta t_{i,k} = t^{due}(k) - t^{rel}(k) \quad &\forall \quad t^{rel}(i) < t^{rel}(k) < t^{due}(k) < t^{due}(i)
\end{aligned} \tag{7.48}$$

Diese Überlappung sollte in die Berechnung der Umschlagskennziffer mit einfließen, somit ergibt sich:

$$U_{i,k} = \frac{B}{n_{i,k} \cdot \Delta t_{i,k} \cdot \sum_{c \in C_{i,k}} |x_{s,c} - x_{z,c}|} \tag{7.49}$$

Die in (7.43) angegebene Zielfunktion behält ihre Gültigkeit.

Bei diesem Ansatz werden Fälle, wie in Abbildung 7.17 dargestellt, gleich behandelt. Abhängig von den Krangrenzen und der Anzahl der eingesetzten Kräne ist

allerdings der obere Fall (d.h. ein Direktumschlag, große Anzahl an Kränen) oder der untere Fall (d.h. zwei Direktumschläge bei kleiner Anzahl Kräne oder großer Bereichsüberlappung) besser.

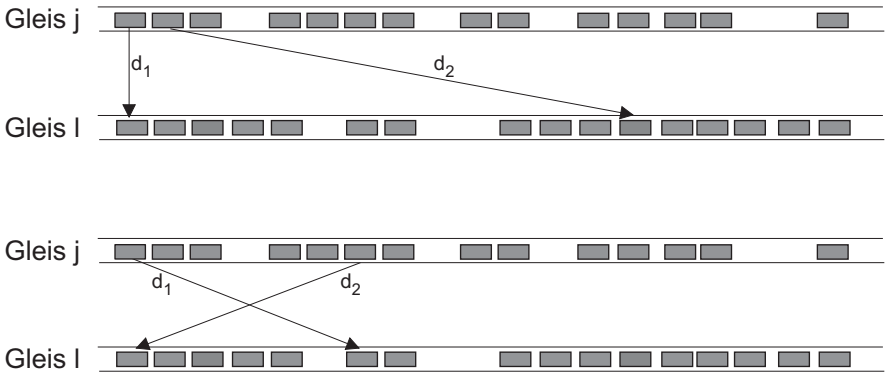


Abbildung 7.17.: Darstellung der Problematik durch Nichtbeachtung der Kranbereiche.

Dieses Problem läßt sich vermeiden, wenn die Anzahl der eingesetzten Kräne und deren Arbeitsbereiche berücksichtigt werden. Damit läßt sich die *mögliche* Anzahl an Direktumschlägen ermitteln, die wiederum in das Optimierungsmodell eingebracht wird.

7.5. Anwendung auf das reale System

Die vorgestellte Modellierung, Formulierung und Implementierung als Constraint Optimization Problem wird im folgenden auf das reale System, das in Abbildung 7.2 dargestellt ist, angewendet. Die Untersuchung kann unterschieden werden in die Bestimmung der optimalen *Konfiguration* des Terminals und dem sich anschließenden optimalen Betrieb. Die Konfiguration definiert die Anzahl der eingesetzten Kräne, FTF sowie die Größe der Bereichsüberschneidungen. Zur systematischen Untersuchung wurde ein Experimenten-Design definiert, das zur statistischen Absicherung auf fünf unterschiedliche Umschlagmatrizen (Replikationen) angewendet wird. Die Umschlagsmatrix wurde mit dem in Anhang B.2 beschriebenen Algorithmus erzeugt. Zugrunde liegt eine Umschlagsmatrix mit 240 Ladeeinheiten, somit werden rund 70% der Container beim Aufenthalt der Züge umgeschlagen. Die untersuchten Parameter sind

- die Anzahl der Kräne (6, 8, 10),
- die Anzahl der FTF (60, 90),
- die Überschneidung der Kranbereiche (0, 1, 2).

Die Bewertung der Ergebnisse erfolgt bei beiden Untersuchungen über die Laufzeit t_{cpu} und den Zielfunktionswert L_{max}. Die Zuweisung der Gleise wurde mit dem in Kapitel 7.4 vorgestellten Verfahren ermittelt. Die Versuche wurden auf einem Pentium Pro Rechner mit 450 Mhz Taktfrequenz und 128 MB Hauptspeicher ausgeführt.

	Fahrplan 1		Fahrplan 2	
Zug	t^{rel}	t^{due}	t^{rel}	t^{due}
1	0 (0)	56 (3360)	0 (0)	42 (2520)
2	8 (480)	64 (3840)	6 (360)	48 (2880)
3	16 (960)	72 (4320)	12 (720)	54 (3240)
4	24 (1440)	80 (4800)	18 (1080)	60 (3600)
5	32 (1920)	88 (5280)	24 (1440)	66 (3960)
6	40 (2400)	96 (5760)	30 (1800)	72 (4320)

Tabelle 7.3.: Fahrplan der Züge, die Einfahrreihenfolge ist gleich der Ausfahrreihenfolge. Alle Züge sind für nur 16 (12) [min] zugleich im Terminal. Die Zeiten sind in [min] und ([sec]) angegeben.

Nach Meyer (1999) wird angenommen, daß die Züge im Abstand von 8, bzw. 6 Minuten in das Teminal einfahren. Die Fahrpläne sind in Tabelle 7.3 gegeben. Es

wird angenommen, daß die Ladeeinheiten gleichverteilt auf den Zügen ankommen und die Zielpositionen ebenfalls wieder gleichverteilt sind. Bei der Untersuchung wird davon ausgegangen, daß die Ladeeinheiten auf Züge des gleichen Bündels umgeschlagen werden. Es ist bei Verspätung eines Zuges nicht möglich, auf das nächste Bündel zu warten.

Die kinematischen Werte für den Mega Hub sind für Katz- und Kranfahrt identisch, lediglich die Fahrzeuge des Sortierspeichers unterscheiden sich:

$$
\begin{aligned}
v_{Kran} &= v_{Katz} = 3\,[\frac{m}{s}] \\
v_{Heben/Senken,beladen} &= 0,75\,[\frac{m}{s}] \\
v_{Heben/Senken,leer} &= 1,5\,[\frac{m}{s}] \\
a_{Kran} &= a_{Katz} = 0,54\,[\frac{m}{s^2}] \\
a_{Heben} &= a_{Senken} = 1\,[\frac{m}{s^2}] \\
v_{LF} &= 3\,[\frac{m}{s}] \\
a_{LF} &= 0,3\,[\frac{m}{s^2}]
\end{aligned}
$$

Bei einer zurückzulegenden Strecke von $l \geq 16,666\ [m]$ geht die Kran- bzw. Katzfahrt in konstante Fahrt über (unter Verwendung von Gleichung (7.5)). Für die Fahrzeuge des Sortierspeichers liegt diese Grenze bei $l \geq 30\ [m]$ reiner Fahrzeit, für die Hub/Senkbewegung bei $l^{voll} = 0,5625\ [m]$ und $l^{leer} = 2,25\ [m]$. Die Hubhöhe über Schienenkante ist mit 10,3 [m] angegeben, d.h. es ergeben sich als Hub- Senkzeiten $t^{auf,voll} = t^{ab,voll} = 14,5\ [sec]$ und $t^{auf,leer} = t^{ab,leer} = 8,36\ [sec]$.

Bei dem realen System sind die Gleise nicht in konstantem Abstand angeordnet, in Tabelle 7.4 ist die entsprechende Distanzmatrix angegeben.

Bei Umschlägen von Kategorie drei wird angenommen, daß der Kran von der Startposition auf direktem Weg zum Längsförderer fährt, somit nur ein Katzspiel nötig ist. Die Handlingsvorgänge sind ebenfalls für jede Ladeeinheit gleich.

	d_{ij} [m]	1	2	3	4	5	6	7	8
Gleis 1	1	0	8,1	13	19,8	31,3	39,05	43,95	52,35
Gleis 2	2		0	4,9	11,7	23,2	30,95	35,85	44,25
Gleis 3	3			0	6,8	18,3	26,05	30,95	39,35
Sortierer 1	4				0	11,5	19,25	24,15	32,55
Sortierer 2	5					0	7,75	12,65	21,05
Gleis 4	6						0	4,9	13,3
Gleis 5	7							0	8,4
Gleis 6	8								0

Tabelle 7.4.: Entfernungsmatrix der realen Anlage, aus Symmetriegründen ist nur die obere Dreiecksmatrix angegeben.

7.5.1. Konfiguration des Terminals

Der Fahrplan 2 kann nicht eingehalten werden, wie in Tabelle 7.6 zu erkennen ist. Interessant ist die gesamte Durchlaufzeit C_{max} aller Aufträge unter der Berücksichtigung der Ausfahrzeit des letzten Zuges, der das Terminal zum Zeitpunkt $t^{due} = 4320[sec]$ verläßt. Durch eine Anpassung des Fahrplanes könnte hier die Verspätung vermieden werden. Weitere Versuche mit Fahrplan 1 haben dies bestätigt. Eine Änderung des Fahrplanes hat allerdings wiederum Auswirkungen auf das Gesamtnetz.

	Ausprägung		
Parameter	-	0	+
Anzahl Kräne, K	6	8	10
Anzahl FTF, f_s	60		90
Bereichsüberlappung, r	0	1	2

Tabelle 7.5.: Die untersuchten Parameter des Experimentendesigns.

In Abbildung 7.18 ist zu sehen, daß die minimale Verspätung bei Verwendung von 10 Kränen, 90 FTF und einer überlappenden Reihe erreicht wird. Der Anstieg der Verspätung bei zwei alternativen Kranreihen, der auch bei Verwendung von sechs Kränen zu erkennen ist, kann durch die Zielfunktion erklärt werden. Die Minimierung der maximalen Verspätung wird in der Regel durch wenige *Engpaß*-Aufträge bestimmt, sie ist nicht kontinuierlich wie MIN C_{max}. Es kann beispielsweise vorkommen, daß einer oder mehrere dieser Engpaßaufträge in dem Überlappungsbereich liegen und von einem Umschlag der Kategorie drei in Kategorie zwei umgewandelt wird. Sind diese Umschläge dem Kran zugewiesen, der auch den

Engpaß-Auftrag auszuführen hat, muß dieser für den L_{max} - bestimmenden Zug *mehr* Zeit für Umschläge aufbringen. Somit verschiebt sich die Abfahrt des Zuges weiter.

	Parameter			Ergebnisse		
Lauf	K	f	r	$\overline{L}_{max}$	$\overline{C}_{max}$	$\overline{t}_{cpu}$
1	6	60	0	577	4109	1532
2	6	60	1	717	4123	104
3	6	60	2	1156	4408	224
4	6	90	0	872	4065	48
5	6	90	1	375	4163	128
6	6	90	2	605	4314	184
7	8	60	0	151	3348	4004
8	8	60	1	280	3513	882
9	8	60	2	143	3564	130
10	8	90	0	130	3327	46
11	8	90	1	230	3411	207
12	8	90	2	238	3383	25
13	10	60	0	Laufzeit > 7200 [sec]		
14	10	60	1	324	3591	1032
15	10	60	2	334	3586	301
16	10	90	0	72	3194	98
17	10	90	1	41	3008	133
18	10	90	2	86	3283	120

Tabelle 7.6.: Versuche und Ergebnisse, zugrunde lag Fahrplan 2.

Wird die Anzahl der FTF reduziert, ergibt sich der in Abbildung 7.19 dargestellte Verlauf. Man vermutet einen Engpaß und damit den Anstieg der Zielfunktionswerte. Teils trifft dies zu, es gibt jedoch Abweichungen. Je mehr Kräne eingesetzt werden, um so stärker ist die Belastung des Sortierspeichers zu Beginn, da zwar von den eingefahrenen Zügen abgeladen, aber noch nicht auf die Zielzüge aufgeladen werden kann. Bei einer Krananzahl von 6 ist der Anstieg bei einer Erhöhung der alternativen Reihen verwunderlich. Die Kranbereiche bei sechs Kränen sind doppelt so groß wie beispielsweise bei einem Terminal mit zwölf Kränen. Zur Abarbeitung der erwähnten zielfunktionsbestimmenden Engpaßaufträge können also große Leerfahrten nötig sein, was sich wiederum ungünstig auf den Zielfunktionswert auswirkt.

Die Nutzung des Sortierspeichers zeigt Abbildung 7.20, im Mittel wird zu 81% der Zeit, die die Fahrzeuge mit Containern belegt sind, gespeichert und zu 19% gefahren. Diese Auswertung bestätigt die Vernachlässigung der Leerfahrt des Sor-

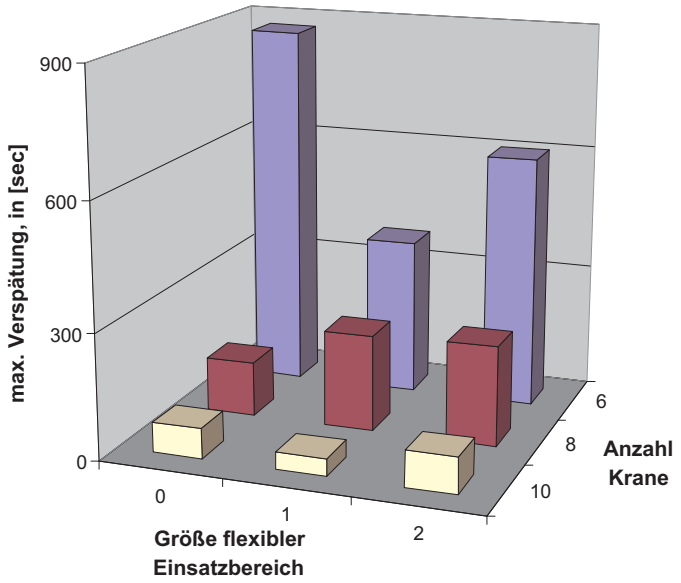


Abbildung 7.18.: Entwicklung des Zielfunktionswertes L_{max} in Abhängigkeit von der Anzahl alternativer Reihen und der Anzahl eingesetzter Kräne bei Verwendung von 90 FTF.

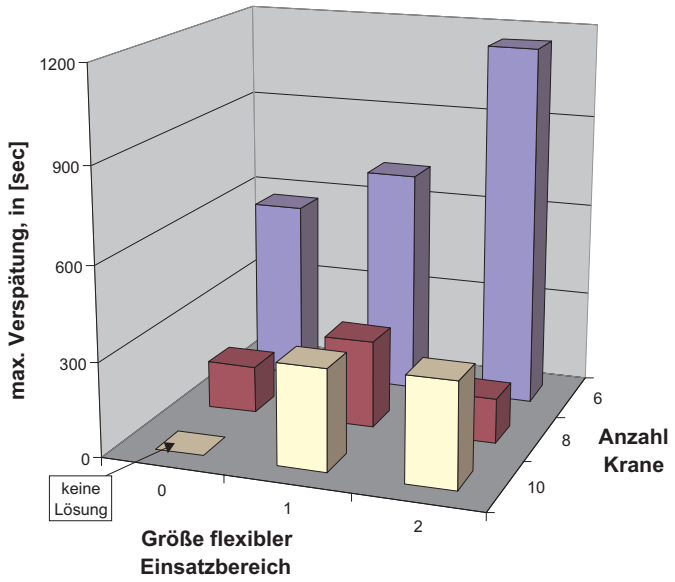


Abbildung 7.19.: Entwicklung des Zielfunktionswertes L_{max} in Abhängigkeit von der Anzahl alternativer Reihen und der Anzahl eingesetzter Kräne bei Verwendung von 60 FTF.

tierspeichers. Sehr deutlich ist die Einfahrt der Züge zu erkennen. Im Abstand von 480 [sec] fährt ein Zug ein, die Anzahl der fahrenden FTF steigt. Wenn der erste Zug abgeladen und die FTF die Zielposition erreicht haben, müssen die Container noch weiter gespeichert werden, da die Zielzüge noch nicht eingefahren sind. Das ist an dem gleichbleibendem Verlauf der speichernden und fahrenden FTF zu erkennen. Ab der Einfahrt von Zug 5 bei 1920 [sec] können die Container nicht mehr sofort auf die FTF verladen werden, die Ab- und Auflade-Vorgänge überlagern sich.

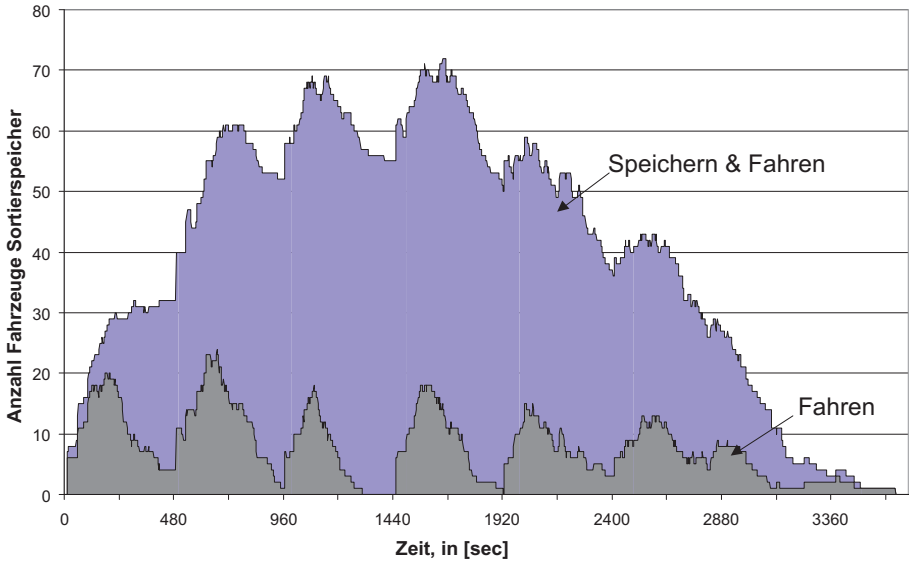


Abbildung 7.20.: Verlauf der Nutzung des Sortierspeichers, aufgegliedert nach Fahren sowie nach Fahren und Speichern. Die zugrundeliegende Konfiguration waren 240 umzuschlagende Container, 8 Kräne, zwei überlappende Reihen und 80 FTF. Die Züge sind nach Fahrplan 1 ein- und ausgefahren.

Die Auslastungen der Kräne, aufgeteilt nach den Anteilen für Leerfahrt, Vollfahrt, Aufnehmen/Absenken und Warten, sind in Abbildung 7.21 dargestellt. Die unterschiedlichen Gesamtzeiten entstehen, da pro Kran vom Umschlagsbeginn des ersten Containers bis zum Umschlagsende des letzten Containers gemessen wird. Die Vollfahrten unterscheiden sich um bis zu $\Delta t^{voll} = 367$ [sec], ähnlich verhalten sich die Leerfahrten. Der Grund liegt wiederum in der Zielfunktion. Wenn die zielfunktionsbestimmenden Aufträge umgeschlagen sind, treten die Voll und Leerfahrten in den Hintergrund. Allerdings ist es bei einem solch komplexen System schwierig, diese Menge von Engpaßaufträgen zu bestimmen.

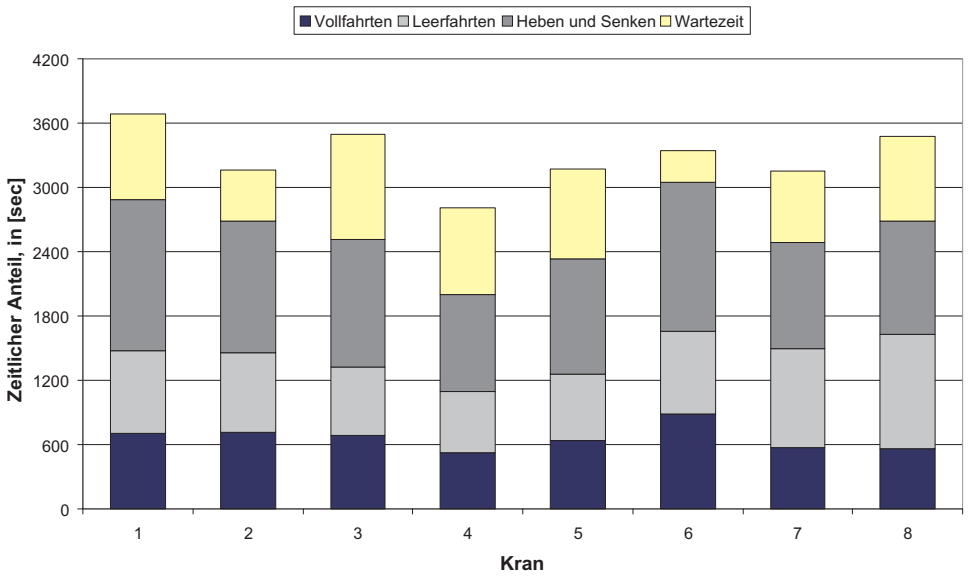


Abbildung 7.21.: Auslastung der Kräne, aufgeteilt nach Zeiten für Leerfahrt, Vollfahrt, Heben/Senken und Warten.

In den Tabellen 7.7 und 7.8 wurde die Konfiguration mit acht, zehn und zwölf Kränen bei Verwendung von Fahrplan 2 untersucht. Die Züge können rechtzeitig ausfahren, die Anzahl der Kräne hat nahezu keinen Einfluß auf das Ergebnis, wenn keine Bereichsüberschneidungen verwendet werden. Bei zunehmender Bereichsüberschneidung wirken sich zusätzliche Kräne negativ aus. Der beste Zielfunktionswert wird bei Verwendung von acht Kränen und zwei Überschneidungsreihen erzielt. Diese Konfiguration sollte für den Betrieb vorgesehen werden.

	Parameter			erste Lösung			optimale Lösung			
Lauf	K	f	r	t_{cpu}	L_{max}	C_{max}	t_{cpu}	L_{max}	C_{max}	#Lsg.
1	8	90	0	4.62	-315	3088	14.94	-637	3404	3
2	8	90	1	24.6	-315	3088	103.47	-676	3365	5
3	*8*	*90*	*2*	62.73	-225	3136	254.36	**-681**	3268	6
4	10	90	0	9.01	1447	4996	27.3	-637	4332	3
5	10	90	1	22.9	-227	3176	98.7	-635	3517	5
6	10	90	2	62.5	-331	3129	253.15	-635	3420	4
7	12	90	0	8.68	117	3520	55.86	-635	3870	7
8	12	90	1	28.29	-87	3349	143.14	-635	3763	5
9	12	90	2	63.71	-146	3298	288.25	-635	3685	5

Tabelle 7.7.: Laufzeitvergleich bei Verwendung von Fahrplan 1. 8 Minuten Aufenthaltszeit ist ausreichend, alle Züge können rechtzeitig abfahren.

	Parameter							#Entsch.-	Speicher-
Lauf	K	f	r	#Var.	$\|\mathcal{C}\|$	$\|\mathcal{O}\|$	$\|\mathcal{R}\|$	punkte	nutzung [KB]
1	8	90	0	1399	2066	466	35	1861	1745.23
1	8	90	1	1756	2720	460	35	3506	1961.15
1	*8*	*90*	*2*	2101	3355	455	35	4896	2300.73
1	10	90	0	1414	2079	471	29	2346	2149.59
1	10	90	1	1657	2533	461	29	3308	1961.15
1	10	90	2	1963	3089	455	29	3266	2228.11
1	12	90	0	1393	2032	464	23	3703	1961.15
1	12	90	1	1582	2394	455	23	3155	1972.93
1	12	90	2	1777	2753	449	23	3546	2133.89

Tabelle 7.8.: Vergleich der COP-spezifischen Werte, wobei $|\mathcal{C}|$ die Anzahl der Constraints, $|\mathcal{O}|$ die Anzahl der Operationen und $|\mathcal{R}|$ die Anzahl der Ressourcen darstellt.

7.5.2. Betrieb des Terminals

Anhand der besten Terminal-Konfiguration wird der optimale Betrieb untersucht, hier werden die folgenden unterschiedlichen Verfahren zur Variablen-Sortierung verglichen:

1. **MinEndMax**: Standardverfahren, das die Variablen mit den kleinsten frühesten Startzeitpunkten und unter diesen die mit dem kleinsten spätesten Endzeitpunkt auswählt. Die Variablen werden damit in ihrer zeitlichen Reihenfolge ausgewählt. Dabei werden Variablen mit kleinen Wertebereichen (gemessen am spätesten Endzeitpunkt) bevorzugt mit Werten belegt.

2. **MinEndMin**: Standardverfahren, das unter den Variablen mit dem kleinsten frühesten Startzeitpunkt die mit dem kleinsten frühesten Endzeitpunkt auswählt.

3. **MinSizeInt**: Standardverfahren, das die Variable mit der geringsten Anzahl an Werten im Wertebereich $|D|$ auswählt.

4. **ORRstatic**: Entspricht dem in Kapitel 5.4 vorgestellten Verfahren, das den Bedarf einer Operation und die Belastung einer Ressource nutzt, um die nächste Variable auszuwählen. Dieses Verfahren wurde auf die vorliegende Anwendung des Mega Hub angepaßt. ORRstatic bedeutet, daß die Reihenfolge einmal vor der Optimierung bestimmt wird.

5. **ORRdynamic**: Bei der dynamischen Variante werden die Bedarfsprofile mehrfach während der Suche erstellt. Für die folgenden Versuche wurde die Berechnung nach jeweils fünf ausgewählten Variablen erneut durchgeführt.

Die Sortier-Verfahren wurden in zwei Varianten getestet: Bei Variante A wird den Operationen, die von zwei verschiedenen Kränen ausgeführt werden können, nach der in Kapitel 7.3.6 vorgestellten Heuristik einer der beiden Kräne zugewiesen und im Anschluß daran die Umschlagsreihenfolge optimiert. Variante B bestimmt die Ressourcenallokation und Umschlagsreihenfolge simultan.

Die Laufzeitergebnisse sind in Tabelle 7.9 dargestellt. Szenario 1 bestand aus einer überlappenden Reihe, 90 FTF, 10 Kränen und Fahrplan 1, Szenario 2 aus zwei überlappenden Reihen, 90 FTF, 12 Kränen, ebenfalls Fahrplan 1. Die Verfahren liefern immer die optimale Lösung. Die einfachen, generischen Standardverfahren haben bei Szenario 1 ein sehr gutes Laufzeitverhalten, bei Szenario 2 dagegen sind ORRstatic und ORRdynamic besser. Erstaunlicherweise wird bei Verwendung der Zuordnungsheuristik bei MinEndMin und MinEndMax in Szenario 2 in der Laufzeitbeschränkung keine Lösung ermittelt. Hier könnte die (recht einfache) Zuweisungsheuristik verbessert werden. Wird keine Zuordnungsheuristik verwendet, steigt die Komplexität aufgrund der parallelen Bestimmung der Ressourcenallokation und Umschlagsreihenfolge an. Vergleicht man die entsprechenden Verfahren ohne die Zuordnungsheuristik, so schneidet das Verfahren ORRdynamic am Besten ab.

Bei dem Vergleich der Anzahl Entscheidungspunkte bei Verwenden von ORRstatic und ORRdynamic in Tabelle 7.10 zeigt sich der größere Aufwand für die Berechnung des Bedarfs-Profiles (ORRdynamic). Bei der Suche werden keine aufwendigen Zwischenlösungen oder Suchbäume gespeichert, daher ist die Speichernutzung sehr gering, was die kombinatorische Komplexität des Problems unterstreicht.

Sortier-	Zuw.		erste Lösung			optimale Lösung			Anz.
Verfahren	Heur.	Szen.	t_{cpu}	L_{max}	C_{max}	t_{cpu}	L_{max}	C_{max}	Lsg.
MinEndMin	mit	1	5.87	-214	3332	30.37	-656	3580	6
MinEndMax	mit	1	7.69	-173	3403	28.02	-656	3522	5
MinEndMin	ohne	1	25.26	-410	3136	105.24	-656	3542	3
MinEndMax	ohne	1	25.37	-368	3208	142.2	-656	3484	5
ChooseMinSizeInt		1	Laufzeit > 3600 [sec]						
ORR static		1	27.3	-410	3595	100.63	-656	4246	4
ORR dynamic		1	27.52	-404	3602	94.03	-656	3743	2
MinEndMin	mit	2	Laufzeit > 3600 [sec]						
MinEndMax	mit	2	Laufzeit > 3600 [sec]						
MinEndMin	ohne	2	69.97	-538	3079	266.11	-656	3267	3
MinEndMax	ohne	2	70.08	-521	3164	282.04	-656	3267	4
ChooseMinSizeInt		2	Laufzeit > 3600 [sec]						
ORR static		2	73.38	-396	3386	276.5	-656	3401	3
ORR dynamic		2	74.37	-566	3244	259.31	-656	3294	2

Tabelle 7.9.: Vergleich der Sortierverfahren, für MinEndMin und MinEndMax wird unterschieden ob die Zuweisungs-Heuristik verwendet wurde oder nicht.

Sortier-	Zuw.-					#Entsch.-	Speicher-
Verfahren	Heur.	#Var.	$\|\mathcal{C}\|$	$\|\mathcal{O}\|$	$\|\mathcal{R}\|$	punkte	nutzung [KB]
MinEndMin	mit	1387	2282	462	29	3230	1815.9
MinEndMax	mit	1387	2282	462	29	2769	1819.82
MinEndMin	ohne	1684	2579	462	29	2239	1945.45
MinEndMax	ohne	1684	2579	462	29	5603	1945.45
ChooseMinSizeInt		Laufzeit > 3600 [sec]					
ORR static		1684	2579	462	29	4323	2338.03
ORR dynamic		1684	2579	462	29	2998	2432.25
MinEndMin	mit	Laufzeit > 3600 [sec]					
MinEndMax	mit	Laufzeit > 3600 [sec]					
MinEndMin	ohne	2161	3454	455	35	2880	2324.28
MinEndMax	ohne	2161	3454	455	35	3598	2332.13
ChooseMinSizeInt		Laufzeit > 3600 [sec]					
ORR static		2161	3454	455	35	4173	2795.38
ORR dynamic		2161	3454	455	35	2920	2716.86

Tabelle 7.10.: Vergleich der COP-spezifischen Werte, wobei $|\mathcal{C}|$ die Anzahl der Constraints, $|\mathcal{O}|$ die Anzahl der Operationen und $|\mathcal{R}|$ die Anzahl der Ressourcen darstellt.

7.6. Zusammenfassung

Das intermodale Terminal-Konzept des Mega Hub stellt für einen optimalen Betrieb hohe Anforderungen an das einzusetzende Steuerungssystem. Die zu lösenden Fragen beinhalten die Bestimmung einer optimalen Umschlagsreihenfolge, die Definition von Kranbereichen, die überlappend sein können, und die Belegung der Gleise. Die unterschiedlichen Umschlagsarten werden analysiert, eine Klassifizierung, die eine Modellierung nach Kapitel 4 ermöglicht, wird vorgestellt. Das komplizierte Problem der reihenfolgeabhängigen Anzahl von Umschlägen wird über die Integration von reihenfolgeabhängigen Übergangszeiten ermöglicht. Um einen Ausgleich der Kranbelastungen zu erreichen, werden die Grenzen der disjunkten Kranbereiche flexibel gestaltet. Die Container in diesen überlappenden Bereichen können alternativ von den angrenzenden Kränen umgeschlagen werden. Für die Belegung der Gleise wird eine Umschlagskennziffer eingeführt, mit deren Hilfe ein quadratisches Zuordnungsproblem formuliert wird.

Die Modellierung wird in ein COP überführt. Die Formulierung wird anschließend auf ein reales System angewendet. Hier werden Untersuchungen zur Konfiguration und zum optimalen Betrieb durchgeführt. Die Ergebnisse zeigen, daß bei einer Konfiguration von 8 Kränen, 90 FTF und 2 überlappenden Containerreihen die minimale Verspätung erzielt werden kann. Die Laufzeiten sind für einen praktischen Einsatz geeignet, hier muß die Neuberechnung einer Umschlagsreihenfolge innerhalb von 5 min (die Zeit von dem Passieren der letzten Blockstrecke bis zur Einfahrt in das Terminal) möglich sein.

8. Ausblick

Die Ausführungen der vorliegenden Arbeit haben gezeigt, daß die Modellierung und Optimierung von mehrstufigen Umschlagsystemen unter Berücksichtigung von praxisrelevanten Randbedingungen komplex, aber machbar ist. Die Ergebnisse der beiden Anwendungen sind sehr vielversprechend, daher ist die Frage nach der Integration von komplexeren Restriktionen oder die Anwendung auf noch größere Problemstellungen zu stellen. Die vorliegene Arbeit kombiniert Entwicklungen auf *logistischem* und *algorithmischem* Gebiet. In diese beiden Richtungen soll daher auch dieser Ausblick aufgeteilt werden. Das Ziel der Lösung von logistischen Problemen bedarf immer mehr einer interdisiziplären Vorgehensweise, daher sind beide Gebiete gleichermaßen interessant und bauen aufeinander auf.

Kann ein Problem der Größe n (in angemessener Zeit) gelöst werden, stellt sich sofort die Frage nach der Lösbarkeit des Problems der Größe $n + 1$. Logistische Systeme haben leider die Tendenz, immer größer, komplexer und damit schwerer beherrschbar zu werden. Eine algorithmische Modellierung und Lösung der Probleme bietet sich an, hier werden die Systeme in Elementen und Beziehungen beschrieben, die Komplexität entsteht durch das Zusammenspiel in dem realen System.

8.1. Logistische Sicht

In der vorliegenden Arbeit wurden wichtige Nebenbedingungen beachtet und in die Modellierung sowie Formulierung integriert. Aufgrund des angemessenen Aggregationsgrades kann die Lösung in annehmbarer Zeit ermittelt werden. Erweiterungspotentiale liegen in der Verfeinerung der Modellierung, die im folgenden für die beiden Anwendungsfälle aufgezeigt wird.

Mehrstufige Kommissioniersysteme

Der zackenartige und teils oszillierende Verlauf der Nutzung der Personalressource könnte durch eine Übergangszeit zwischen den Stufen vermieden werden. Die Integration des Planers bei dem Vorschlag einer Personalplanung ist wichtig, hier können Interaktionsmöglichkeiten geschaffen werden. Die Methode des Constraint Logic Programming läßt eine solche Interaktion über die Definition von zusätzlichen Constraints zu.

In der vorliegenden Anwendung befinden sich die Kommissionierplätze in räumlicher Nähe zueinander. Große reale Distributionssysteme haben häufig ein sehr inhomogenes Produktspektrum, was zu einer (durchaus sinnvollen) Ansammlung von unterschiedlichen Lagerbereichen und Kommissionierkonzepten führt. Die Synchronisation der Materialströme im Versandbereich ist jedoch eine allgemeine Anforderung. Weitere Kommissionier- und auch Packbereiche können in das Modell integriert werden. Interessant ist hier speziell die Integration von Kommissionierbereichen mit statischer Bereitstellung. Die Anwendung von Verfahren zur Lösung der entstehenden Travelling Salesman Probleme in den Kommissionierbereichen müssen in die Constraint Satisfaction Ansätzen zur Sicherstellung eines synchronisierten Materialflusses des Gesamtsystems integriert werden.

Umschlagterminal Schiene-Schiene

Verbesserungen des Terminal-Konzeptes können in die Modellierung aufgenommen werden. Beispielsweise könnte Umschlagzeit gespart werden, wenn ein Kran analog zur Synchronfahrt eines Regalbediengerätes während des Katzspieles in Richtung der Zielposition fährt. Zusätzliche Constraints sind nötig, um diese Erweiterung zu modellieren.

Interessant ist die Integration des Modells in die Untersuchung des Gesamtnetzes. Wird ein Terminal-Verbund analysiert, sind Fragen nach einem optimalen Fahrplan und der Konfiguration des Gesamt-Netzes zu beantworten. Hier löst das vorliegende Modell wichtige Teilfragen. Bislang wurde nur ein Bündel von Zügen betrachtet, d.h. alle Container mußten in dem Bündel umgeschlagen werden. Eine Erweiterung auf mehrere Terminals und mehrere Bündel macht die Mitnahme eines Containers zu einem späteren Zeitpunkt auf einem späteren Zug möglich. Als weitere Zielgrößen sind die Machbarkeit der erzeugten Umschlagmatrix, der Servicegrad und der unterstellte Fahrplan zu integrieren.

Das Terminal-Konzept ist nicht an den Schiene-Schiene Umschlag gebunden, prototypische Anwendung werden auch für den Hafen-Bereich entwickelt. Manuell bediente Kräne entladen seeseitig die Schiffe und beladen landseitig die Lkw oder

Züge, die komplette Sortierung und Feinverteilung im Terminal wird von automatischen Kränen und einer großen Sortieranlage übernommen. Der Sortierspeicher wurde in dem vorliegenden Modell vereinfacht. Bei einer Anwendung im Hafen hat der Sortierspeicher eine weitaus komplexere Aufgabe, so daß die Frage nach dem optimalen Einsatz der FTF gelöst werden muß. Die Erweiterung des Modells zur Optimierung einer solchen Anlage ist eine herausfordernde Fragestellung.

8.2. Algorithmische Sicht

Das Konzept des Constraint Logic Programming bietet die Möglichkeit, Nebenbedingungen sehr realitätsnah zu integrieren und zugleich eine ausreichende Geschwindigkeit bei der Bestimmung von Lösungen zu erzielen. Durch den allgemeinen Ansatz ist es auf zahlreiche Problemstellungen anwendbar. Wie sich in dieser Arbeit, speziell in Kapitel 7, gezeigt hat, beeinflussen andererseits angepaßte Lösungsverfahren, hier die Sortierung der Variablen, die Laufzeit positiv. Darin liegt noch Potential für weitere Forschungs-Arbeiten. Das Verstehen der Struktur eines Problems und hieraus abgeleitete Verfahren zur Sortierung von Variablen oder allgemein Suchverfahren bietet ein weites Anwendungsfeld für wissenschaftliche Arbeiten. Die Integration von puffernden Ressourcen in die Auswahlverfahren sollte erweitert werden. Die Variablen-Sortierverfahren bei Verwendung von alternativen Ressourcen, bei denen zusätzlich reihenfolgeabhängige Übergangszeiten zu beachten sind, können noch verbessert werden.

Die Verfahren stoßen ab einer bestimmten Problemgröße an Laufzeitgrenzen. Hier kann die Kombination aus langfristigen, aggregierten und kurzfristigen, detaillierten Verfahren interessant sein. Beispielsweise kann bei einem komplexen Netzwerk alternativer Ressourcen die Warteschlangentheorie genutzt werden, um langfristige Kapazitätsaussagen zu treffen und die Verteilung der Ströme auf die alternativen Ressourcen in gewissen Grenzen festzulegen. Bei diesem Horizont sind Verteilungen, d.h. aggregierte Aufträge interressant. Im kurzfristigen Bereich sollen nun die Aufträge eingeplant werden, die Verteilung der Aufträge auf die Ressourcen wird in das Optimierungsmodell in Form von Constraints integriert. Die Ergebnisse der detaillierten Stufe können wiederum zur Anpassung der aggregierten Stufe genutzt werden.

8.3. Allgemeine Erweiterungen

Die Unterstützung des Planers bei der Modellierung komplexer Systeme kann deutlich verbessert werden. Eine mathematische Formulierung ist für Wissenschaftler gut geeignet, um schnell einen Sachverhalt aufzunehmen und zu verstehen. Der Planer wird jedoch eher abgeschreckt. Im Bereich der praxisnahen Modellierung von Systemen besteht noch Forschungsbedarf. Modellierungsunterstüzende Systeme wie OPL (Van Hentenryck und Lustig (1999)), AMPL (Fourer, Gay und Kernighan (1999)) etc. leisten einen Beitrag durch Kapselung der Lösungskomponente, doch muß der Planer sich mit der zugrundeliegenden Theorie auseinandergesetzt haben. Die Tools leisten keine Unterstüzung bei der Entscheidung einer angemessenen Aggregation oder/und Disaggregation von Problemen. Die Vereinigung des Entwicklers und Modellierers wird teils kritisch gesehen, siehe z.B. Gass (1990), was die Bedeutung der modellierungsunterstützenden Systemen unterstreicht.

Während der Bestimmung einer Lösung werden zahlreiche Informationen, beispielsweise in Form von (suboptimalen) Zwischenergebnissen, erzeugt, die im Verlauf der Optimierung nicht weiter verwendet werden. Hier können wertvolle Erkenntnisse abgeleitet werden, wie alternative (suboptimale) Ergebnisse, die dem Planer vorgeschlagen und interaktiv verändert werden. Die Nutzung dieses Wissens, um zukünftige Entscheidungen schneller zu treffen, ist ein interessantes Forschungsgebiet, siehe auch Shanahan und Southwick (1989).

In der vorliegenden Arbeit wurden vorausgesetzt, daß die Aufträge a-priori vorhanden sind. Diese Annahme ist für die Untersuchung der Effekte von Kapazitätsbeschränkungen und der Unterscheidung Beachtung/Nichtbeachtung der Batchzuweisung in Kapitel 6 realistisch. In einem realen System ist der Auftragseingang jedoch kontinuierlich, also wird eine rollierende Planung nötig. Diese muß nicht in vorgegebenen Zeitabständen (z.B. eine Stunde), sondern kann auch kontinuierlich erfolgen. Die vorgestellten Modelle können um eine solche Planung erweitert werden, hier muß allerdings darauf geachtet werden, daß durch die rollierende Planung eine gewisse Nervosität in dem System erzeugt werden kann.

Soll auf Ereignisse reagiert werden, bietet sich die Anwendung von Regelsystemen an. Komplexe Systeme führen oft zu ebenso komplexen und schwer vorhersagbaren Regelsystemen. Die Ableitung von Regeln aus dem mathematischen Modell ist eine interessante Aufgabe. Sehr vielversprechend erscheint die Anwendung von Ansätzen aus der mathematischen Kontrolltheorie zu sein. Hier wird versucht, einen (langfristig) optimalen Betriebspunkt zu halten, Steuerungsmechanismen sorgen für eine Anpassung der Entscheidungsregeln bei Abweichung vom Betriebspunkt.

9. Zusammenfassung

Das Ziel der vorliegenden Arbeit ist die Modellierung und Optimierung von praxisrelevanten, mehrstufigen Umschlagsystemen. Hier besteht eine wichtige Anforderung in der Maximierung des Servicegrades unter Verzicht auf den Aufbau von Lagerbeständen entlang der Wertschöpfungskette. Beispiele für solche Systeme sind das in Kapitel 6 behandelte Kommissioniersystem und das in Kapitel 7 untersuchte Umschlagsterminal.

Die Schwierigkeit bei der Optimierung solcher Systeme liegt in der Berücksichtigung der zahlreichen Nebenbedingungen. Werden die Systeme vereinfacht oder werden zu viele Annahmen getroffen, sind die Lösungen nicht mehr auf die Realität übertragbar. Andererseits birgt eine zu detaillierte Betrachtung die Gefahr, daß nur ein sehr spezielles Problem gelöst wird, und die Übertragung auf andere Praxisfälle damit nur schwer möglich ist.

Das den untersuchten mehrstufigen Umschlagsystemen zugrunde liegende Optimierungsproblem kann beschrieben werden durch

- eine mehrstufige Bearbeitung,
- einen beliebigen Fluß der Aufträge (oder synonym Fördereinheiten) durch das System,
- reihenfolgeabhängige Übergangszeiten,
- zeitabhängige Kapazitäten der Ressourcen,
- die Verwendung von alternativen Ressourcen,
- den Einsatz kapazitativ beschränkter Puffer, die zusätzlich Bearbeitungsfunktionen übernehmen,
- den Einsatz kapazitativ beschränkter flexibler Pool-Ressourcen,
- vorgegebene Starttermine und gewünschte Endtermine.

Die beiden interessantesten Probleme aus dieser Aufzählung sind die kapazitativ beschränkten Puffer und die flexiblen Personal-Ressourcen. Zur Entkopplung der Stufen werden Puffer verwendet, die neben einer gegebenen Kapazität auch (z.B. FTF) Transport- oder sonstige Bearbeitungsfunktionen übernehmen. Dieser Fall tritt bei dem untersuchten Umschlagsterminal Schiene-Schiene auf, das in Kapitel 7 beschrieben wird.

Personal ist eine knappe und wertvolle Ressource und sollte daher möglichst effektiv eingesetzt werden. Ein flexibler Personaleinsatz wird durch die Bildung von Pools erreicht. Die Ressourcen der einzelnen Stufen können nach Bedarf auf das Personal des Pools zurückgreifen. Sowohl die fixierten (Kommissionier- oder Packplätze) als auch die flexiblen Ressourcen (Personalpool) können den Engpaß des Systems darstellen. Diese Anforderungen lassen sich nicht mit Standardverfahren lösen, sind für die angesprochene praktische Relevanz aber unbedingt nötig.

In den vergangenen Jahren konnten zahlreiche Probleme der kombinatorischen Optimierung mit Constraint Satisfaction Ansätzen gelöst werden. Die Modellierung mit Constraints ist problemnah und läßt sich sehr gut auf die untersuchten Probleme anwenden. Weiterhin stehen zahlreiche generische Konzepte zur Einschränkung des Suchraumes und Beeinflussung der Suche zur Verfügung. In Kapitel 5 werden die theoretischen Grundlagen geschaffen, um das Problem als *Constraint Optimization Problem* zu formulieren. Zur Lösung der untersuchten Probleme waren spezielle Erweiterungen nötig, die vorgestellt und integriert werden.

In dem ersten Anwendungskapitel wird der COP-Ansatz auf ein *mehrstufiges Kommissioniersystem* angewendet. Hier sind spezielle Reihenfolge-Constraints sowie die Beachtung/Nicht-Beachtung einer Batchzuweisung zu beachten. Anhand von Szenarien und zufällig erzeugten Problemen werden das Laufzeitverhalten und die Zielfunktionswerte bewertet und diskutiert. Die Laufzeiten der Berechnungen sind auch bei großen Problemen akzeptabel. Ein interessantes Ergebnis ist der geringe Einfluß der Beachtung/Nicht-Beachtung der Batchzuweisung auf die maximale Verspätung. Die Probleme weisen ein ausgeprägtes Phasen-Übergangs-Verhalten auf. Hier war das Ziel die Bestimmung von sehr guten Lösungen und nicht der Beweis der Optimalität. Begrenzte Puffer und ein begrenzter Personal-Pool zeigen, daß nicht ausschließlich eine Ressource den Engpaß des Systems darstellt. Vielmehr bilden Kombinationen von Ressourcen einen Engpaß, der sich dynamisch ändert. Mit einer flexible Personaleinsatzplanung läßt sich das System optimal betreiben.

Die Anwendung auf ein *Umschlagsystem Schiene-Schiene*, dem Mega Hub, schließt die Arbeit ab. Hier sind alternative Zuordnungen von Containern zu Kränen, reihenfolgeabhängige Leerfahrten und eine Kombination aus puffernder

und bearbeitender Ressource zu beachten. Das entwickelte Modell wird zur Untersuchung der Konfiguration des Terminals und zur Optimierung der operativen Abläufe verwendet. Es zeigt sich, daß auch hier unterschiedliche Engpässe interagieren, aufbauend auf den Untersuchungen wird eine Empfehlung für eine Konfiguration gegeben. Bei Veränderung des Fahrplanes, d.h. bei (fast) gleichzeitigem Einfahren der Züge, können die Ressourcen besser ausgelastet werden. Die Untersuchung des dynamischen Sortierspeichers zeigt, daß dieser hauptsächlich zum Speichern der Container und nur zu ca. 20 Prozent zum Fahren eingesetzt wird. Im Laufzeitvergleich der unterschiedlichen Verfahren zur Sortierung der Variablen schneiden die einfachen, generischen Verfahren sehr gut ab, werden jedoch bei der Hälfte der untersuchten Szenarien von den weiterentwickelten Verfahren ORRstatic und ORRdynamic übertroffen.

Literatur

Alicke, K. und D. Arnold (1997a). KOMAX - Optimierung im Kommissionierbereich. *Hebezeuge und Fördermittel 37*(9), S. 343–345.

Alicke, K. und D. Arnold (1997b). Simulation and Optimization of a Pull-Strategy in the Order-Picking Area of a Distribution Warehouse. In: *11^{th} European Simulation Multiconference, Istanbul*, S. 369–374. SCS International.

Alicke, K. und D. Arnold (1998). Optimierung von mehrstufigen Umschlagsystemen. *Fördern und Heben 8*(10), S. 769–772.

Alicke, K., D. Arnold, J. Nienhaus und K.-P. Franke (1999). Modelling and Simulation of the Intermodel Terminal Mega Hub. In: *HMS 99 - International Workshop on Harbour, Maritime & Logistics Modelling and Simulation, Genua*.

Alicke, K., D. Arnold und J. Schweitzer (1998). Einsatz von Simulation bei der Optimierung von zweistufigen Lager- und Kommissioniersystemen. *Erfahrungen aus der Zukunft, Tagungsband der 8. ASIM-Fachtagung, Berlin*, S. 31–41.

Alicke, K. und B. Faisst (1999). Modellierung und Optimierung von Kommissioniersystemen in der Praxis. *VDI-Seminar Optimierte Kommissioniersysteme*.

Alicke, K. und M. ten Hompel (1999). *OEM - Das Objekt/Ebenenmodell der Lagerverwaltung*. Praxiswissen.

Allen, J. (1983). Maintaining Knowledge About Temporal Intervals. *Communications of the ACM 26*, S. 832–843.

Arnold, D. (1995). *Materialflußlehre*. Vieweg.

Arnold, D. und B. Rall (1996a). Neues Umschlag- und Transportsystem für den kombinierten Verkehr. *Logistik im Unternehmen 10*(7/8), S. 59–61.

Arnold, D. und B. Rall (1996b). Neues Umschlagsystem für Güterverkehrszentren im Vergleich mit aktuellen Konzepten. *VDI-Berichte* (1274), S. 197–208.

Ashour, S. (1972). *Sequencing Theory*. Springer, Berlin.

Azadivar, F. (1986). Maximization of the Throughput of a Computerized Automated Warehousing System under System Constraints. *International Journal of Production Research 24* (3), S. 551–566.

Baker, K. R. (1974). *Introduction to Sequencing and Scheduling*. John Wiley & Sons.

Baptiste, P., C. le Pape und W. Nuijten (1995). Constraint-Based Optimization and Approximation for Job-Shop Scheduling. In: *AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems*.

Bauer, R. (1998). Innovative Linear Motor-Based Transfer Technology allows intelligent Container Handling. In: *Proceedings of EURNAV 98, Hannover*.

Beck, J. C. (1999). *Texture Measurements as a Basis for Heuristic Commitment Techniques in Constraint-Directed Scheduling*. Dissertation, University of Toronto.

Blazewicz, J., W. Domschke und E. Pesch (1996). The Job Shop Scheduling Problem: Conventional and New Solution Techniques. *European Journal of Operational Research 93*, S. 1–33.

Bozer, Y. A. und J. A. White (1984). Travel-Time Models for Automated Storage/Retrieval Systems. *IEE Transacations 16* (4), S. 329–338.

Brailsford, S. C., C. N. Potts und B. M. Smith (1999). Constraint Satisfaction Problems: Algorithms and Applications. *European Journal of Operational Research 119*, S. 557–581.

Breitinger, S. und C. Lock (1994). Improving Search for Job-Shop Scheduling with CLP(FD). In: *Programming Language Implementation and Logic Programming*, S. 277–291.

Brockhaus (1983). *Naturwissenschaften und Technik*. F. A. Brockhaus, Wiesbaden.

Carlier, J. und E. Pinson (1989). An Algorithm for Solving the Job-Shop Problem. *Management Science 35*, S. 164–176.

Caseau, Y. und F. Laburthe (1994). Improved CLP Scheduling with Task Intervals. In: *Proceedings of the 11th International Conference on Logic Programming*, S. 369–383.

Caseau, Y. und F. Laburthe (1995). Disjunctive Scheduling with Task Intervals. Forschungsbericht Liens-95-25, Laboratoire d'Informatique d L'Ecole Normale Superieure.

Chen, H., C. Chengbin und J.-M. Proth (1998). Cyclic Scheduling of a Hoist with Time Window Constraints. *IEEE Transactions on Robotics and Automation 14*(1), S. 144–152.

Cheng, C.-C. und S. F. Smith (1997). Applying Constraint Satisfaction Techniques to Job Shop Scheduling. *Annals of Operations Research 70*, S. 327–357.

Cormier, G. und E. A. Gunn (1992). A Review of Warehouse Models. *European Journal of Operational Research 58*, S. 3–13.

Daniels, R. L., J. L. Rummel und R. Schantz (1998). A Model for Warehouse Order Picking. *European Journal of Operational Research 105*, S. 1–17.

Dechter, R. (1990). Enhancement Schemes for Constraint Processing: Backjumping, Learning and Cutset Decomposition. *Artificial Intelligence 41*, S. 273–312.

Derrick, T. C. und M. Pegman (1998). Overview of Constraint-based Scheduling. *http://www.cs.unh.edu/ccc/archive/constraints/archive/app-sched.txt*.

Derstroff, M. (1995). *Mehrstufige Losgrößenplanung mit Kapazitätsbeschränkungen*. Physica, Heidelberg.

Dörrsam, V. (1999). *Materialflußorientierte Leistungsanalyse einstufiger Produktionssysteme*. Dissertation, Universität Karlsruhe.

Eisenführ, F. und M. Weber (1993). *Rationales Entscheiden*. Springer, Berlin.

Elsayed, E. A., M. K. Lee, S. Kim und E. Scherer (1993). Sequencing and Batching Procedures for Minimizing Earliness and Tardiness Penalty of Order Retrievals. *International Journal of Production Research 31*(3), S. 727–738.

Fischer, H. und G. L. Thompson (1963). *Probablistic Learning Combinations of Local Job-Shop Scheduling Rules*. In Muth und Thompson (1963).

Fourer, R., D. M. Gay und B. W. Kernighan (1999). *AMPL - A Modeling Language For Mathmatical Programming*. Thomson Publishing.

Fox, M. S. (1983). *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. Dissertation, Carnegie-Mellon University.

Fox, M. S. (1994). *ISIS: A Retrospective*, Chapter 1, S. 3–27. In Zwieben und Fox (1994).

Franke, K.-P. (1997). Mega-Drehscheibe für den kombinierten Verkehr. *EIA Symposium Europe Towards Intermodal Transport*.

Franke, K.-P. und G. Häffner (1996). Automatisierung der Umschlagsknoten als Beitrag der Industrie, Transportketten wirtschaftlicher zu machen. *VDI-Berichte* (1274), S. 181–195.

Freuder, E. C. (1982). A Sufficient Condition for Backtrack-Free Search. *Journal of the ACM 29*(1), S. 24–32.

Gass, S. I. (1990). Model World: Danger, Beware the User as Modeler. *Interfaces 20*(3), S. 60–64.

Goetschalckx, M. und H. D. Ratliff (1988). Order Picking in an Aisle. *IIE Transactions 20*(1), S. 53–62.

Goetschalckx, M. und H. D. Ratliff (1990). Shared Storage Policies based on the Duration Stay of Unit Loads. *Management Science 36*(9), S. 1120–1132.

Goldratt, E. und J. Cox (1998). *The Goal: A Process of Ongoing Improvement* (2 Aufl.). North River Press.

Goltz, H.-J. (1995). Reducing Domains for Search in CLP(FD) and its Application to Job-Shop Scheduling. In: *Principles and Practice of Constraint Programming - CP '95*, S. 549–562.

Graves, S., A. H. G. Rinnooy Kan und P. Zipkin (1993a). *Logistics of Production and Inventory*, Volume 4 of *Handbooks in Operations Research and Management Science*. North-Holland.

Graves, S., A. H. G. Rinnooy Kan und P. Zipkin (1993b). *Sequencing and Scheduling: Algorithms and Complexity*, Chapter 9, S. 445–521. Volume 4 of *Handbooks in Operations Research and Management Science* Graves, Rinnooy Kan und Zipkin (1993a).

Greiling, M. (1998). *Verbesserung der Produktionslogistik durch Losgrößenharmonisierung*. Dissertation, Universität Karlsruhe.

Gudehus, T. (1973). *Grundlagen der Kommissioniertechnik*. Giradet.

Günther, H.-O. (1989). *Produktionsplanung bei flexibler Personalkapazität*. Poeschel.

Günther, H.-O. und H. Tempelmeier (2000). *Produktion und Logistik*. Springer, Berlin.

Gutenschwager, K., S. Spieckermann und S. Voß (1998). Order Sequencing in an Automated Warehouse System. *Proceedings of the Industrial Engineering Research Conference*, S. 1–10.

Hall, N. G., M. E. Posner und C. N. Potts (1997). Preemptive Scheduling with Finite Capacity Input Buffers. *Annals of Operations Research 70*, S. 399–413.

Hall, N. G., M. E. Posner und C. N. Potts (1998). Scheduling with finite Capactity Output Buffers. *Operations Research 46*, S. 84–97.

Hall, R. W. (1985). What's So Scientivic about MS/OR. *Interfaces 15*(2), S. 40–45.

Hall, R. W. (1993, July). Distance Approximations for Routing Manaual Pickers in a Warehouse. *IIE Transactions 25*(4), S. 76–87.

Hanen, C. (1994). Study of a NP-hard Cyclic Scheduling Problem: The Recurrent Job-Shop. *European Journal of Operational Research 72*(1), S. 82–101.

Harris, F. W. (1913). How Many Parts to Make at Once. *Factory: The Magazine of Management 10*(2), S. 135–136, Reprint, *Operations Research, 38*(6), 1990, S. 947–950.

Hillier, F. S. und G. J. Lieberman (1995). *Introduction to Operations Research.* McGraw-Hill.

Hopp, W. J. und M. L. Spearman (1996). *Factory Physics.* McGraw-Hill.

Jünemann, R. (1989). *Materialfluß und Logistik.* Springer, Berlin.

Khosla, I. (1995). The Scheduling Problem where Multiple Machines compete for a Common Local Buffer. *European Journal of Operational Research 84*, S. 330–342.

Kumar, V. (1992). Algorithms for Constraint Satisfaction Problems: A Survey. *AI Magazine 13*(1), S. 32–44.

Le Page, P. und G. Barras (1997). Application of ILOG Solver in fast Intermodal Transshipment. White Paper, http://www.ilog.com.

Le Pape, C. (1994). *Scheduling as Intelligent Control of Decision-Making and Constraint Propagation*, Chapter 3, S. 67–97. In Zwieben und Fox (1994).

Lei, L. und T.-J. Wang (1991). The Minimum Conmmon Cycle Algorithm for Cycle Scheduling of Two Material Handling Hoists with Time Window Constraints. *Management Science 37*(12), S. 1629–1639.

Mackworth, A. K. (1987). *Constraint Satisfaction*, S. 205–211. In Shapiro Shapiro (1987).

Mayoh, B. (1993). *Constraint Programming.* Springer, Berlin.

Meyer, P. (1999). *Entwicklung eines Simulationsprogramms für Umschlagterminals des kombinierten Verkehrs.* Dissertation, Universität Hannover.

Muth, J. und G. L. Thompson (1963). *Industrial Scheduling.* Prentice Hall.

Nadel, B. A. (1989). Constraint Satisfaction Algorithms. *Computational Intelligence 5*, S. 188–224.

Ng, W. (1996). A Branch and Bound Algorithm for Hoist Scheduling of a Circuit Board Production Line. *International Journal of Flexible Manufacturing Systems 8*, S. 45–65.

N.N. (1995). Analyse und Bewertung der mittel- bis langfristigen Perspektiven einer Schienenvernetzung von KV-Umschlagbahnhöfen unter Berücksichtigung neuer Ansätze für die Produktionsgestaltung im Rahmen eines zentralen Hub-and-Spoke-Systems für den kombinierten Verkehr. Schlußbericht 90433/94, Verkehrsministerium.

N.N. (1997). Container Transportsysteme der Zukunft. Schlußbericht zum F&E-Vorhaben Nr. 18S0071A, BMBF.

N.N. (1999). Müntefering sucht Leithammel - Messe Eurocargo '99. *Logistik Heute 21* (5), S. 22–26.

Nuijten, W. (1994). *Time and Resource Constraint Scheduling - A Constraint Satisfaction Approach*. Dissertation, Technische Universiteit Eindhoven.

Osman, I. H. und J. P. Kelly (1996a). *Meta-Heuristics: An Overview*, Chapter 1, S. 1–21. In Osman und Kelly (1996b).

Osman, I. H. und J. P. Kelly (1996b). *Meta-Heuristics: Theory & Application*. Kluwer Academic Publishers.

Panwalkar, S. S. und W. Iskander (1977). A Survey of Scheduling Rules. *Operations Research 25* (1), S. 45–61.

Phillips, L. und P. Unger (1976). Mathematical Programming Solution of a Hoist Scheduling Problem. *AIIE Transactions 8* (2), S. 219–225.

Pinedo, M. (1995). *Scheduling: Theory, Algorithms and Systems*. Prentice-Hall.

Pountain, D. (1995). Constraint Logic Programming. *Byte 20* (2), S. 159–160.

Prosser, P. (1993). Hybrid Algorithms for the Constraint Satisfaction Problem. *Computational Intelligence* (9), S. 268–299.

Ratliff, H. D. und A. S. Rosenthal (1983). Order-Picking in a Rectangular Warehouse: A Solvable Case of the Travelling Salesman Problem. *Operations Research 31* (3), S. 507–521.

Rodosek, R. und M. Wallace (1998). One Model and Different Solvers for Hoist Scheduling Problems. *Unpublished working paper, Presentation at INFORMS 1998, Montreal*.

Roodbergen, K. J. und R. d. Koster (1998). Routing Order Pickers in a Warehouse with Multiple Cross Aisles. Forschungsbericht, Rotterdam School of Management, Erasmus University Rotterdam.

Rosenwein, M. B. (1996). A Comparison of Heuristics for the Problem of Batching Orders for Warehouse Selection. *International Journal of Production Research 34*(3), S. 657–664.

Ruben, R. A. und F. R. Jacobs (1999). Batch Construction Heuristics and Storage Assignment Strategies for Walk/Ride and Pick Systems. *Managment Science 45*(4), S. 575–596.

Rumbaugh, J., M. Blaha und W. Premerlani (1993). *Objektorientiertes Modellieren und Entwerfen*. Hanser.

Sadeh, N. (1991). *Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling*. Dissertation, Carnegie Mellon University.

Sadeh, N. (1994). *Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler*, Chapter 4, S. 99–135. In Zwieben und Fox (1994).

Sadeh, N. und M. S. Fox (1996). Variable and Value Ordering Heuristics for the Job Shop Scheduling Constraint Satisfaction Problem. *Artificial Intelligence 86*, S. 1–41.

Sadeh, N., K. Sycara und Y. Xiong (1995a). Backtracking Techniques for the Job Shop Scheduling Constraint Satisfaction Problem. *Artificial Intelligence 76*, S. 455–480.

Sadeh, N., K. Sycara und Y. Xiong (1995b). Intelligent Backtracking Techniques for Job Shop Scheduling. In: *Proceedings of the third International Conference on Principles of Knowledge Representation and Reasoning*, S. 14–23.

Schiex, T. (1997). An Overview of Valued CSP and Related Results. In: *EUFIT '97, Aachen*, S. 942–946.

Shanahan, M. und R. Southwick (1989). *Search, Interference and Dependencies in Artificial Intelligence*. John Wiley & Sons.

Shapiro, S. (Hrsg.) (1987). *Encyclopedia of Artificial Intelligence*. Wiley.

Simonis, H. und T. Cornelissens (1995). Modelling Producer/Consumer Constraints. In: *Principles and Practice of Constraint Programming - CP '95*, S. 449–462.

Stallman, R. und G. Sussman (1977). Forward Reasoning and Dependency Directed Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence*.

Van Hentenryck, P. und I. Lustig (1999). *The OPL Optimization Programming Language*. MIT Press.

Varnier, C., A. Bachelu und P. Baptiste (1997). Resolution of the Cyclic Multi-Hoists Scheduling Problem with Overlapping Partitions. *Information Systems and Opterational Research 35*(4), S. 1–16.

VDI 3590 (1994). *Kommissioniersysteme*. VDI-Verlag.

Wallace, M. (1993). *Applying Constraints for Scheduling*, Chapter 2.5. In Mayoh (1993).

Yih, Y. (1994). An Algorithm for Hoist Scheduling Problems. *International Journal of Production Research 32*(3), S. 501–516.

Yoon, C. S. und G. P. Sharp (1996). A Structured Procedure for Analysis and Design of Order Pick Systems. *IIE Transactions 28*, S. 379–389.

Zwieben, M. und M. S. Fox (1994). *Intelligent Scheduling*. Morgan Kaufman.

A. Formelzeichen

Es wurde versucht, die Bezeichnung der Formelzeichen an die umgangssprachliche Bezeichnung anzupassen. Das hat in wenigen Fällen zu Mehrfachnutzungen geführt, die aber aus dem Kontext sehr einfach abzuleiten sind.

Indizes

a	Index der Artikel
c	Index der Constraints
i	Index
j	Index der Aufträge (Jobs)
k	Index Kommissionierstufe
o, p	Laufindizes von Operationen
p	Index Packerei
pos	Index der Positionen eines Auftrages
r	Index der Ressourcen
r_p	Index der Pool-Ressourcen
t	Index der Touren
v	Index Versand
z	Index der Züge

Mengen

$\mathcal{A}$	Menge aller Artikel
C	Menge der Constraints
D	Menge der Wertebereiche der Variablen
F_i	Menge der Container, die Umschlag-Kategorie i entsprechen
X	Menge der Constraint-Variablen
$\mathcal{J}$	Menge aller Aufträge (Jobs)
$\mathcal{O}$	Menge aller Operationen
$\mathcal{POS}$	Menge aller Positionen
$\mathcal{R}$	Menge aller Ressourcen
$\mathcal{RS}$	Menge aller Mengen von Ressourcen
$\mathcal{T}$	Menge aller Touren

Sonstige Bezeichner

σ	Schedule (Belegungsplan)
a	Beschleunigung
aj	In einem Auftrag enthaltene Artikel
ap	In einem Auftrag enthaltene Positionen
as	Artikel einer Position
at	In einer Tour enthaltene Aufträge
b_{kp}	Puffer zwischen Kommissionierbereich und Packerei
b_{pv}	Puffer zwischen Packerei und Versand
cp	Kapazität einer Ressource
$d_{i,j}$	(euklidischer) Abstand von i nach j
est	Frühest möglicher Startzeitpunkt einer Operation
ect	Frühest möglicher Fertigstellungszeitpunkt einer Operation
fr	definiert eine Menge von Ressourcen auf denen eine Operation ausgeführt werden kann
h	Personal-Pool
l	Länge
lst	Spätest möglicher Startzeitpunkt einer Operation
lct	Spätest möglicher Fertigstellungszeitpunkt einer Operation
pt	Bearbeitungszeit einer Operation auf einer Ressource
p	Wahrscheinlichkeit
q	Zu kommissionierende Menge eines Artikels (einer Position)
ra	Definiert für eine Operation eine Ressource
sz	Ressourcen-Bedarf einer Operation
t^{rel}	frühest möglicher Startzeitpunkt
t^{due}	gewünschter Fertigstellungszeitpunkt
t^{set}	Übergangszeit zwischen zwei Operationen
t_s	Startzeitpunkt
t_e	Endzeitpunkt
x, y	Koordinaten
y_{ij}	Binärvariable, wird eins, wenn Zug i auf Gleis j einfährt
C	Gesamtanzahl der umzuschlagenden Container
C_{max}	Durchlaufzeit aller Aufträge
G	Anzahl der Gleise
H	Ende des Untersuchungszeitraumes
I	(zeitliches) Intervall
K	Anzahl der Kräne
L_{max}	maximale Verspätung aller Aufträge
O_i^j	Operation i von Auftrag j

P	Position
SD	Daten, um ein MUP zu definieren
$U_{i,j}$	Umschlagskennziffer
Z^r, Z^k	Kern- und Randbereiche des Einsatzbereiches eines Kranes

B. Erzeugung der Szenarien

B.1. Mehrstufige Kommissioniersysteme

Um die vorgestellten Verfahren zu testen, wurden Szenarien mit Hilfe der in Kapitel 6.5.1 beschriebenen Parameter automatisch erzeugt. Die geänderten Parameter sind in den jeweiligen Kapiteln angegeben.

Parameter, die während der Untersuchung nicht geändert wurden, sind die Bearbeitungszeiten an den einzelnen Stufen, die sich immer auf eine Einheit beziehen (d.h. ein Artikel, eine Position oder ein Packstück). Weiterhin wurden die Übergangszeiten zwischen unterschiedlichen und gleichen Artikeln im Kommissionierbereich und dem Rücktransport in das Lager nicht geändert:

$$pt(kom) = 1\ [ZE]$$
$$pt(pack) = 2\ [ZE]$$
$$pt(versand) = 2\ [ZE]$$
$$t_v = 5\ [ZE]$$
$$t_g = 0\ [ZE]$$
$$t_{k,r,k} = 30\ [ZE]$$

Nach Algorithmus 3 wird die Struktur der Probleme definiert. Die mit ZUFALL(x,y) erzeugten Zufallszahlen seien gleichverteilt in dem Intervall (x,y).

Algorithmus 3 Erzeugung der Szenarien für die Untersuchung der mehrstufigen Kommissioniersysteme

Erzeuge $AnzArtikel$ unterschiedliche Artikel: $\mathcal{A}$
Weise Artikel Klassen A oder C zu
Erzeuge $AnzTouren$ Touren: $\mathcal{T}$
for $t \in \mathcal{T}$ **do**
  $t^{due}(t) = \text{ZUFALL}(MinDue, MaxDue)$
  $t_j = \text{ZUFALL}(1, MaxAuftraege)$
  Erzeuge t_j Aufträge und weise sie Tour t zu
  for $j \in at(t)$ **do**
    $j_p = \text{ZUFALL}(1, MaxPositionen)$
    Erzeuge j_p Positionen und weise sie Auftrag j zu
    for $pos \in ap(pos)$ **do**
      Wähle Artikel a gemäß AC-Verteilung: $as(pos) = a$
      $q(pos) = \text{ZUFALL}(1, MaxMenge)$
    end for
  end for
end for

B.2. Mehrstufiges Umschlagsystem

Die Umschlagsmatrix wird mit Algorithmus 4 erzeugt. Die umzuschlagenden Ladeeinheiten werden gleichmäßig auf die ankommenden und abfahrenden Züge verteilt. Die mit ZUFALL(x,y) erzeugten Zufallszahlen seien gleichverteilt in dem Intervall (x,y). Ein Umschlagsauftrag $j \in \mathcal{J}$ entspricht einer Ladeeinheit, für die Start- und Zielposition zu $P_s(j)$ und $P_z(j)$ angegeben werden können.

Algorithmus 4 Generierung der Umschlagsmatrix

Erzeuge die Menge der Start- $\mathcal{P}_s$ und Ziel-Positionen $\mathcal{P}_z$.
for $j \in \mathcal{J}$ **do**
  Wähle zufällige Startposition $P_s(j) \in \mathcal{P}_s$
  Wähle zufällige Zielposition $P_z(j) \in \mathcal{P}_z | P_z(j) \neq P_s(j)$
  $\mathcal{P}_s \backslash \{P_s(j)\}$
  $\mathcal{P}_z \backslash \{P_z(j)\}$
end for

C. Aspekte der Integration und Implementation

In dieser Arbeit wurden ein Constraint Optimization Problem formuliert um optimale Umschlagsreihenfolgen zu ermittelen. Die untersuchten mehrstufigen Umschlagsysteme sind ein Schiene-Schiene Terminal und ein mehrstufiges Kommissioniersystem. In diesem Kapitel sollen Aspekte der Integration der Ansätze als Optimierungsmodul in ein rechnergestütztes Planungs- und Steuerungssystem gegeben werden.

Die wichtigsten Anforderungen an ein solches Optimierungsmodul sind

- **Beachtung der Randbedingungen des Systems**: In Kapitel 4 wurde die Modellierung von wichtigen Randbedingungen dargestellt, diese sind
  - Beachtung von unterschiedlichen Kapazitäten der Ressourcen, die auch zeitabhängig sein können,
  - Beachtung von Pufferkapazitäten,
  - Modellierung von bearbeitenden und puffernden Ressourcen,
  - Integration von flexiblen Pool-Ressourcen, etwa zur Modellierung von Personal,
  - Alternative Zuordnung von Aufträgen zu Ressourcen,
  - Beachtung von reihenfolgeabhängigen Übergangszeiten,
  - Integration von unterschiedlichen Zielfunktionen.

  Die problemnahe Modellierung von Constraints erlaubt die Kombination aus einer sehr detaillierten und aggregierten Modellierung.

- **Integration in rollierende Planung**: Die hier untersuchten Systeme sind zu Beginn leer. Wird allerdings aufgrund einer Störung oder Verspätung eine erneute Planung nötig, sind Umschlags-Aufträge bereits abgeschlossen,

andere werden gerade bearbeitet, etc. Dieser Zustand muß bei der Planung berücksichtigt werden. Über die Modellierung als Constraint Optimization Problem ist diese Anforderung einfach zu erfüllen. Ein Teil der Variablen ist bereits mit Werten belegt, die Suche kann von diesem Punkt starten.

Ist die Bestimmung der Reihenfolgen abgeschlossen, und ein Zug fährt beispielsweise verspätet in das Terminal ein, so kann durch Anwendung der Konsistenz-Algorithmen überprüft werden, ob die ermittelte Reihenfolge verändert werden muß.

- **Optimieren nach unterschiedlichen Zielfunktionen**: Die Zielfunktion wird dem Constraint Satisfaction Problem als Constraint hinzugefügt. Somit ist die Modellierung von sehr unterschiedlichen Zielfunktionen möglich.

- **Schnelle Lösungsbestimmung**: Die Laufzeituntersuchungen haben gezeigt, daß die Probleme schnell gelöst werden können. Ein dedizierter Rechner zur Lösung der Optimierungsprobleme sollte für einen praktischen Betrieb vorgesehen werden, um die zeitkritischen Optimierungsläufe nicht zu behindern.

Die Klassenbibliotheken Solver 4.31 und Scheduler 4.3 der Firma ILOG wurden genutzt, um die Optimierungsprobleme zu modellieren und die Verfahren zu implementieren. Die Bibliotheken liegen in der Sprache C++ vor und stellen Variablen, die Modellierung von einfachen Constraints, Prüfen von Konsistenzbedingungen und die Baumsuche zur Bestimmung einer Lösung zur Verfügung.

Zur strukturierten Implementation wurden für die beiden Optimierungsprobleme die folgenden Klassen definiert:

- Mehrstufige Kommissioniersysteme
  - Artikel
  - Auftrag
  - Packstück
  - Position
  - Tour
- Umschlagsterminal Schiene-Schiene
  - Container
  - Auftrag
  - Kran
  - Sortierspeicher

In den Klassen sind die benötigten Attribute und die Constraint Variablen abgelegt. Die angegebenen Konsistenzbedingungen, sowie die Sicherstellung der Konsistenz und die Lösungssuche nach den beschriebenen Bedarfsprofilen wurde durch Überladen der vorgegebenen Klassen implementiert.